**Л.Е. Еременко**

**РАЗРАБОТКА ПРИЛОЖЕНИЯ СРЕДСТВАМИ  
MSVISUALFOXPRO 9.0**

***Учебно-практическое пособие***

Рязань 2014

MSVisualFoxPro – мощнаясреда программирования для разработки приложений с графический интерфейсом пользователя. Являясь средой программирования, в то же время VisualFoxProимеет множество возможностей по работе с базами данных. Поэтому закономерно использование VisualFoxProдля разработки приложений, обрабатывающий информацию, организованную в виде баз данных.

В учебно-практическом пособии описана процедура разработки подобного приложения.Поскольку информационной основой приложения является база данных, то в пособии приведен пример и ее проектирования, а так же реализация разработанной базы данных средствами MSVisualFoxPro 9.0. В пособии подробно описаны функции будущего приложения, приведены примеры разработки объектов проекта VisualFoxPro, реализующих эти функции. В приложении приводятся многочисленные примеры разработки разнообразных форм, отчетов, запросов, процедур, позволяющих вести обработку информации. Описана процедура разработки меню приложения, а так же получение исполнимого файла, как завершающего этапа разработки приложения.

В пособии приведен минимальный объем теоретических сведений, и излагаемый материал представлен преимущественно в виде практических заданий, описывающих действия по разработке объектов. В целом содержание материала предусматривает возможность его изучения даже пользователямb, практически не знакомымb с VisualFoxPro.

В заключительном разделе пособия представлены примеры типичных ошибок, которые могут возникнуть при работе с VisualFoxPro, и даны рекомендации поих странению.

Пособие может быть использовано студентами специальностей группы 230000, а также любыми лицами, изучающими разработку приложений средствами MSVisualFoxPro 9.0.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_Toc403034010)

[1. ПЛАНИРОВАНИЕ ПРОЕКТИРОВАНИЯ БАЗЫ ДАННЫХ 6](#_Toc403034011)

[1.1. Анализ предметной области 6](#_Toc403034012)

[1.2. Определение задач (функций) приложения 6](#_Toc403034013)

[2. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ 7](#_Toc403034014)

[2.1. Концептуальное проектирование базы данных 7](#_Toc403034015)

[2.1.1. Определение сущностей 7](#_Toc403034016)

[2.1.2. Определение связей между сущностями 7](#_Toc403034017)

[2.2. Логическое проектирование базы данных 7](#_Toc403034018)

[2.2.1. Преобразование сущностей 7](#_Toc403034019)

[2.2.2. Преобразование связей 8](#_Toc403034020)

[2.2.3. Задание первичных ключей, определение атрибутов сущностей 8](#_Toc403034021)

[2.2.4. Приведение модели базы данных к первой нормальной форме 9](#_Toc403034022)

[2.2.5. Приведение модели базы данных ко второй нормальной форме 11](#_Toc403034023)

[2.2.6. Приведение модели базы данных к третьей нормальной форме 13](#_Toc403034024)

[2.3. Физическое проектирование базы данных 15](#_Toc403034025)

[2.3.1. Физическое описание модели базы данных 15](#_Toc403034026)

[2.3.2. Создание базы данных средствами MS Visual FoxPro 9.0 17](#_Toc403034027)

[ЗАДАНИЕ 1. Создание базы данных в проекте MSVisualFoxPro 9.0 18](#_Toc403034028)

[3. ЗАПОЛЕНИЕ И ПРОСМОТР ТАБЛИЦ БАЗЫ ДАННЫХ В РЕЖИМЕ BROWSE 23](#_Toc403034029)

[3.1. Заполнение и просмотр таблиц 23](#_Toc403034030)

[ЗАДАНИЕ 2. Заполнение и просмотр таблиц базы данных 23](#_Toc403034031)

[3.2. Просмотр и отбор содержимого таблиц с помощью команды Browse 25](#_Toc403034032)

[ЗАДАНИЕ 3. Отображение и отбор содержимого таблиц 26](#_Toc403034033)

[3.3. Поиск данных в таблице 27](#_Toc403034034)

[ЗАДАНИЕ 4. Поиск данных в таблице 28](#_Toc403034035)

[4. РАЗРАБОТКА ФОРМ ДЛЯ РАБОТЫ С БАЗОЙ ДАННЫХ 29](#_Toc403034036)

[4.1. Создание простейших форм 30](#_Toc403034037)

[ЗАДАНИЕ 5. Создание простейших форм ввода данных 31](#_Toc403034038)

[4.2. Применение конструктора форм 35](#_Toc403034039)

[ЗАДАНИЕ 6. Редактирование формы с помощью конструктора форм 36](#_Toc403034040)

[ЗАДАНИЕ 7. Создание формы с помощью конструктора форм 39](#_Toc403034041)

[4.3. Размещение на форме элемента ComboBox 41](#_Toc403034042)

[Задание 8. Размещение в форме ComboBox и его настройка с помощью Builder 42](#_Toc403034043)

[Задание 9. Размещение в форме ComboBox и его настройка с помощью панели Properties 45](#_Toc403034044)

[4.4. Размещение в форме объекта Grid 45](#_Toc403034045)

[ЗАДАНИЕ 10. Создание формы с объектом Grid 48](#_Toc403034046)

[ЗАДАНИЕ 11. Изменение Grid с помощью панели Properties 52](#_Toc403034047)

[ЗАДАНИЕ 12. Изменение управляющего элемента столбца Grid 54](#_Toc403034048)

[ЗАДАНИЕ 13. Создание вычисляемого столбца в Grid 56](#_Toc403034049)

[4.5. Размещение на форме вычисляемых полей 56](#_Toc403034050)

[ЗАДАНИЕ 14. Осуществление простых вычислений на форме 57](#_Toc403034051)

[ЗАДАНИЕ 15. Программное изменение данных столбца Grid 60](#_Toc403034052)

[ЗАДАНИЕ 16. Вычисление суммы данных столбца Grid на форме 63](#_Toc403034053)

[ЗАДАНИЕ 17. Программное управление объектами формы 66](#_Toc403034054)

[4.6. Размещение на форме объекта CommandButton 70](#_Toc403034055)

[ЗАДАНИЕ 18. Разработка простейших форм поиска данных 70](#_Toc403034056)

[5. РАЗРАБОТКА ОТЧЕТОВ 73](#_Toc403034057)

[5.1. Общие сведения об отчетах. Рекомендации по созданию отчетов 73](#_Toc403034058)

[5.2. Создание простых отчетов с помощью мастера 75](#_Toc403034059)

[ЗАДАНИЕ 19. Создание простейшего отчета с помощью мастера 75](#_Toc403034060)

[5.3. Создание отчетов с группировкой с помощью мастера 79](#_Toc403034061)

[ЗАДАНИЕ 20. Создание простого отчета с группировкой 80](#_Toc403034062)

[5.4. Создание отчетов с итоговыми вычислениями 81](#_Toc403034063)

[ЗАДАНИЕ 21. Создание простого отчета с итоговыми вычислениями 82](#_Toc403034064)

[ЗАДАНИЕ 22. Создание простого отчета с группировкой и итоговыми вычислениями 83](#_Toc403034065)

[5.5. Использование конструктора для работы с отчетами 84](#_Toc403034066)

[ЗАДАНИЕ 23. Использование конструктора для настройки внешнего вида отчета 85](#_Toc403034067)

[5.6. Создание отчетов на основе нескольких таблиц 90](#_Toc403034068)

[ЗАДАНИЕ 24. Создание многотабличного отчета 90](#_Toc403034069)

[5.7. Фильтрация данных отчета 96](#_Toc403034070)

[ЗАДАНИЕ 25. Привязка отчета к текущей записи формы 96](#_Toc403034071)

[5.7. Размещение вычисляемых полей в отчете 97](#_Toc403034072)

[ЗАДАНИЕ 26. Размещение вычисляемых полей в отчете 99](#_Toc403034073)

[6. СОЗДАНИЕ ЗАПРОСОВ И ЛОКАЛЬНЫХ ПРЕДСТАВЛЕНИЙ 103](#_Toc403034074)

[6.1. Общие сведения о запросах и представлениях. Создание запросов и представлений 103](#_Toc403034075)

[ЗАДАНИЕ 27. Создание запросов на основе одной таблицы 105](#_Toc403034076)

[ЗАДАНИЕ 28. Создание локального представления на основе нескольких таблиц 110](#_Toc403034077)

[6.2. Вычисления в запросах и представлениях 113](#_Toc403034078)

[ЗАДАНИЕ 29. Создание вычисляемых полей в представлениях (запросах) 113](#_Toc403034079)

[ЗАДАНИЕ 30. Выполнение итоговых вычислений в представлении или запросе 115](#_Toc403034080)

[ЗАДАНИЕ 31. Использование представления для создания отчета 116](#_Toc403034081)

[7. РАЗРАБОТКА СРЕДСТВ ОБРАБОТКИ ДАННЫХ НА ОСНОВЕ ОПИСАНИЯ ПРЕДМЕТНОЙ ОБЛАСТИ И ПОТРЕБНОСТЕЙ ПОЛЬЗОВАТЕЛЯ 120](#_Toc403034082)

[ЗАДАНИЕ 32. Поиск информации о клиентах-должниках 120](#_Toc403034083)

[ЗАДАНИЕ 33. Применение фильтрации к форме 127](#_Toc403034084)

[ЗАДАНИЕ 34. Создание отчета о договорах проката, оформленных за период 129](#_Toc403034085)

[8. РАЗРАБОТКА МЕНЮ ПРИЛОЖЕНИЯ 137](#_Toc403034086)

[8.1. Общие сведения о меню приложения. Структура меню 137](#_Toc403034087)

[8.2. Подготовка к созданию меню 138](#_Toc403034088)

[ЗАДАНИЕ 34. Настройка путей к объектам 139](#_Toc403034089)

[8.3. Создание справочной системы для пользователя 139](#_Toc403034090)

[ЗАДАНИЕ 35. Использование всплывающих подсказок и строки состояния 140](#_Toc403034091)

[ЗАДАНИЕ 36. Разработка диалоговых окон 140](#_Toc403034092)

[ЗАДАНИЕ 37. Создание формы справочных сведений 141](#_Toc403034093)

[8.4. Создание меню с помощью конструктора меню MenuDesigner 144](#_Toc403034094)

[ЗАДАНИЕ 38. Создание меню приложения 145](#_Toc403034095)

[9. РАЗРАБОТКА ОСНОВНОЙ ПРОГРАММЫ ПРОЕКТА И КОМПИЛЯЦИЯ ПРИЛОЖЕНИЯ 149](#_Toc403034096)

[ЗАДАНИЕ 39. Разработка формы идентификации пользователя 149](#_Toc403034097)

[ЗАДАНИЕ 40. Разработка основной программы проекта 150](#_Toc403034098)

[ЗАДАНИЕ 41. Компиляция приложения 151](#_Toc403034099)

[ПРИЛОЖЕНИЕ 1 Некоторые сообщения об ошибках и их описание 154](#_Toc403034100)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 157](#_Toc403034101)

# ВВЕДЕНИЕ

В современном обществе практически во всех его сферах широко используются автоматизированные информационные системы. На кассе практически любого магазина установлена соответствующая информационная система, при помощи которой оформляются продажи товаров, изменяется их количество в наличии, формируется чек, выдаваемый покупателю. На складе каждого магазина (или просто на складе какой-либо продукции) тоже присутствует информационная система, при помощи которой можно оформить поступление товаров, списание товаров (и их отгрузку при необходимости). Это примеры из сферы торговли. А сфера услуг? В какую бы организацию не обращался человек, наверняка он столкнется там с соответствующей информационной системой, которая позволит сформировать заказ на оказание услуг, в котором будут отображены и сведения о заказчике, и перечень услуг, и их стоимость, и срок выполнения заказа. Примеров таких организаций очень много: и фотоателье, и автосервисы, и химчистки, и мастерские по изготовлению на заказ, и организации по выдаче в аренду. Конечно, не исключено, что в ряде организаций все процедуры выполняются вручную, но это, несомненно, уменьшает скорость оформления заказов (квитанций, договоров и т.д.) и увеличивает вероятность ошибок при различных расчетах. Поэтому преимущества использования автоматизированных информационных систем очевидны. И основные различия между такими информационными системами – это так называемая информационная составляющая (сведения о товарах, услугах, изделиях и т.д.) и среда разработки системы. Информационная составляющая и является теми данными, сбором и обработкой которых занимается автоматизированная система. В целях адекватного отображения изменяющегося реального мира и удовлетворения информационных потребностей пользователей эти данные должны быть организованы в базы данных.

Каждая информационная система в зависимости от ее назначения имеет дело с той или иной частью реального мира, которую принято называть предметной областью системы. Выявление предметной области — это необходимый начальный этап разработки любой информационной системы. Именно на этом этапе определяются информационные потребности всей совокупности пользователей будущей системы, которые, в свою очередь, предопределяют содержание ее базы данных и предоставляемые функции.

В пособии будет пошагово описан пример разработки приложения на основе базы данных для учета дисков в DVD-прокате. В настоящее время услуга проката DVD-дисков получает широкое распространение, поскольку несет в себе массу плюсов. Регулярно на дисках выходят новые игры, фильмы, программы. И не всегда есть возможность их купить, да и стоит ли. Далеко не всегда возникает желание пересматривать фильмы или пользоваться устаревшими играми и программами.

DVD-прокат предоставляет широкий перечень дисков различных категорий и жанров, причем в прокат обращается большое количество клиентов, желающих взять диск. Поэтому перед сотрудниками проката встает большое количество задач: обновление сведений об имеющихся дисках, хранение данных о клиентах, оформление договоров на прокат дисков, определение дисков, для которых истек срок проката, и определение клиентов-должников. Список обязанностей сотрудника DVD-проката можно продолжать и дальше. Вывод один – работа сотрудника требует от него внимательности, сосредоточенности и способности выполнять множество рутинных и однообразных операций.

Поэтому автоматизация работы DVD-проката является необходимым и перспективным процессом.

Разработка будет вестись по следующему плану:

1. Планирование проектирования базы данных.
2. Проектирование базы данных.
3. Разработка компонент приложения.
4. Компиляция приложения.

# 1.ПЛАНИРОВАНИЕ ПРОЕКТИРОВАНИЯ БАЗЫ ДАННЫХ

## 1.1. Анализ предметной области

DVD-прокат занимается выдачей в прокат дисков различного содержания: фильмы, игры, музыка, тв-передачи, электронные книги и т.д. По каждому диску имеется такая информация, как номер, категория и жанр, год выпуска, а так же стоимость проката за сутки и величина залога. Стоимость проката за сутки составляет 10% от величины залога. В базе данных также хранятся сведения обо всех арендаторах и оформленных договорах проката дисков. Вся информация, хранящаяся в базе данных, периодически дополняется, редактируется и сохраняется.

Диски могут быть выданы на срок не более 14 дней. Если клиент берет сразу несколько дисков, то они могут быть взяты только на один срок. Если диски берутся на разные сроки, то для каждого срока оформляется отдельный договор. Оплата проката выполняется при получении клиентом дисков. При оплате с каждого клиента берется залог в размере стоимости диска и сумма проката диска. При возврате дисков сумма залога возвращается клиенту. Если клиент возвращает диск раньше срока, то стоимость проката пересчитывается, и разница выплачивается клиенту. Если диск не возвращен в срок, то сотрудник может связаться с соответствующим клиентом, чтобы предупредить его о завершении срока проката. Если диск остается у клиента дольше срока, указанного в договоре, то сумма проката диска пересчитывается.

Клиент приходит в DVD-прокат и может выбрать диск, представленный на витрине, или выбрать конкретный интересующий диск (при этом сотрудник может провести поиск этого диска). После того, как пользователь определяется с выбором диска (дисков), узнает стоимость проката дисков, начинается оформление выдачи. Сначала сотрудник проката регистрирует клиента в базе пользователей, затем составляет договор, в котором будут указаны все выданные в прокат диски со стоимостью проката каждого из них и общей стоимостью проката. В договоре определяется срок, когда диски должны быть возвращены. Распечатанный договор выдается клиенту, и возврат дисков будет осуществляться по этому договору.

## 1.2. Определение задач (функций) приложения

* регистрация дисков, формирование каталога дисков;
* ведение базы данных клиентов;
* оформление договоров на прокат дисков;
* печать договоров;
* учет дисков;
* определение выручки, полученной от оплаты договоров.
* регистрация сотрудников;
* поиск дисков в базе данных;
* оформление и печать договоров на прокат дисков;
* закрытие договоров на прокат;
* расчет стоимости проката;
* определение просроченных договоров;
* поиск диска по различным критериям: по тематике, по году выпуска, по названию;
* формирование отчета по дискам, находящимся у клиентов;
* определение списков «должников».

# 2. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

## 2.1. Концептуальное проектирование базы данных

### 2.1.1. Определение сущностей

На основе анализа предметной области выделим следующие сущности:

* договор;
* диск;
* сотрудник;
* клиент.

### 2.1.2. Определение связей между сущностями

Представим информационную модель задачи в виде диаграммы **сущность–связь** (рис. 1).

Диск

Договор

Клиент

Сотрудник

Рис.1. Диаграмма **сущность–связь**

В данной диаграмме используются следующие связи. Между сущностями **Диск** и **Договор** используется связь **многие-ко-многим**. Это означает, что один диск может присутствовать в нескольких договорах, а в одном договоре может быть указано несколько дисков.

Между сущностями **Сотрудник** и **Договор** используется связь **один-ко-многим**. Это означает, что один сотрудник может заключить несколько договоров, но в одном договоре может быть указан только один сотрудник.

Между сущностями **Договор** и **Клиент** используется связь **один-ко-многим**. Это означает, что один клиент может заключить несколько договоров, но один договор оформляется только на одного клиента.

## 2.2. Логическое проектирование базы данных

Концептуальнаямодельданныхсостоитизрядакомпонентов: сущностей,связей,атрибутов. Припереходе креляционнойсхемебазы данныхкаждыйизэтихкомпонентовдолженбытьпроанализировани, еслиэтоокажетсянеобходимым,тодажеипреобразован.Изменения, вносимыевпроцессепреобразования,должныбытьтакими,чтобыих результатполностьюотвечалтребованиям,выдвигаемымреляционной моделью данных.

Таким образом, фаза логического проектирования предполагает следующие действия:

* преобразованиеконцептуальноймоделиданныхвлогическую модель, в результате которого будет определена схема реляционной модели данных;
* проверкамоделиспомощьюконцепцийпоследовательной нормализации.

### 2.2.1. Преобразование сущностей

Каждой сущности ставится в соответствие отношение.

Каждыйатрибутсущностистановитсяатрибутомотношения, которому приписывают тип данных и свойство допустимости/недопустимости для него значения NULL (не определен).

Первичный ключ сущности становится первичным отношения. Атрибуты, входящие в первичный ключ, получают свойство обязательности и уникальности.

Концептуальная модель содержит 4 сущности. В виду отсутствия связи 1:1 в реляционной модели будет по крайней мере 4 отношения.

### 2.2.2. Преобразование связей

Преобразование связи производится согласно значениям ее характеристик: типа связи по мощности (один-ко-многим, один-к-одному, многие-ко-многим) и класса принадлежности.

**Класс принадлежности** – показатель обязательности участия каждого экземпляра сущности в связи.

Обязательный класс принадлежности связи показывает, что каждый экземпляр сущности ДОЛЖЕН участвовать в связи. Необязательный класс принадлежности допускает наличие экземпляров сущности, не участвующих в связи.

Проанализируем связи между сущностями.

**Диск – Прокат**, N:M

Связь M:N будет преобразована за счет введения дополнительной сущности со схемой: СОСТАВ (код\_проката, код\_диска, сумма\_за\_диск).

**Прокат – Сотрудник**, 1:N

Каждый договор проката должен быть оформлен одним сотрудником, т.е. многосвязная сущность**Прокат**характеризуется обязательным классом принадлежности. Поэтому две сущности будут представлены двумя отношениями.

**Прокат – Клиент**, 1:N

Каждый договор проката должен быть оформлен одним сотрудником, т.е. многосвязная сущность **Прокат** характеризуется обязательным классом принадлежности. Поэтому две сущности будут представлены двумя отношениями.

Таким образом, получим эквивалентный вариант диаграммы **сущность-связь** (рис. 2).

Диск

Договор

Клиент

Сотрудник

Состав

Рис.2. Диаграмма **сущность–связь** после исключения связи **многие-ко-многим**

### 2.2.3. Задание первичных ключей, определение атрибутов сущностей

Для каждой сущности определим атрибуты, которые будут храниться в базе данных. Результат представлен в табл. 1.

Таблица 1. Атрибуты и первичные ключи сущностей информационной модели

| **Сущность** | **Первичный ключ** | **Атрибуты** |
| --- | --- | --- |
| Договор | код\_договора | код\_договора  код\_клиента  код\_сотрудника  дата\_проката  дата\_оконч\_дог  дата\_оконч\_факт  длительность\_дог  длительность\_факт  сумма  сумма\_за\_сутки  сумма\_залога |
| Состав |  | код\_договора  код\_диска  сумма\_за\_диск |
| Клиент | код\_клиента | код\_клиента  ФИО\_клиента  тел\_клиента  адрес\_клиента  паспорт\_клиента |
| Сотрудник | код\_сотрудника | код\_сотрудника  ФИО\_сотрудника  тел\_сотрудника |
| Диск | код\_диска, код\_жанра, код\_категории | код\_диска  название  год\_выпуска  залог  цена\_сутки  код\_жанра  жанр  код\_категории  категория  наличие |

### 2.2.4. Приведение модели базы данных к первой нормальной форме

Модель базы данных находится в первой нормальной форме, если для каждого ееотношения выполняются следующие условия:

* в отношении отсутствуют повторяющиеся атрибуты;
* каждый атрибут отношения неделим.

Согласно требованиям первой нормальной формы необходимо преобразовать атрибуты **ФИО\_клиента**, **паспорт\_клиента** и **адрес\_клиента** в сущности **Клиент**и атрибут **ФИО\_сотрудника** в сущности **Сотрудник** так, чтобы получить неделимые атрибуты.

Приведем таблицу сущностей и атрибутов в первой нормальной форме (табл. 2).

Таблица 2. Атрибуты и первичные ключи сущностей в первой нормальной форме

| **Сущность** | **Первичный ключ** | **Атрибуты** |
| --- | --- | --- |
| Договор | код\_договора | код\_договора  код\_клиента  код\_сотрудника  дата\_проката  дата\_оконч\_дог  дата\_оконч\_факт  длительность\_дог  длительность\_факт  сумма  сумма\_за\_сутки  сумма\_залога |
| Состав |  | код\_договора  код\_диска  сумма\_за\_диск |
| Клиент | код\_клиента | код\_клиента  фам\_клиента  имя\_клиента  отч\_клиента  тел\_клиента  гор\_клиента  ул\_клиента  дом\_клиента  корп\_клиента  кв\_клиента  серия\_паспорта  номер\_паспорта |
| Сотрудник | код\_сотрудника | код\_сотрудника  фам\_сотрудника  имя\_сотрудника  отч\_сотрудника  тел\_сотрудника |
| Диск | код\_диска, код\_жанра, код\_категории | код\_диска  название  год\_выпуска  залог  цена\_сутки  код\_жанра  жанр  код\_категории  категория  наличие |

Приведем диаграмму взаимосвязей между атрибутами сущностей в первой нормальной форме (рис. 3).

|  |
| --- |
| **Диск** |
| код\_диска |
| название |
| код\_жанра |
| жанр |
| код\_категории |
| категория |
| год\_выпуска |
| залог |
| цена\_сутки |
| наличие |
|  |
|  |

|  |
| --- |
| **Договор** |
| код\_договора |
| код\_клиента |
| код\_сотрудника |
| дата\_проката |
| дата\_оконч\_дог |
| дата\_оконч\_факт |
| длительность\_дог |
| длительность\_факт |
| сумма |
| сумма\_за\_сутки |
| сумма\_залога |
|  |
|  |
|  |
|  |
| **Состав** |
| код\_договора |
| код\_диска |
| сумма\_за\_диск |

|  |
| --- |
| **Клиент** |
| код\_клиента |
| фам\_клиента |
| имя\_клиента |
| отч\_клиента |
| тел\_клиента |
| гор\_клиента |
| ул\_клиента |
| дом\_клиента |
| корпус\_клиента |
| кв\_клиента |
| серия\_паспорта |
| номер\_паспорта |
|  |
|  |
|  |
| **Сотрудник** |
| код\_сотрудника |
| фам\_сотрудника |
| имя\_сотрудника |
| отч\_сотрудника |
| тел\_сотрудника |

Рис.3.Диаграмма взаимосвязей между атрибутами сущностей в первой  
нормальной форме

### 2.2.5. Приведение модели базы данных ко второй нормальной форме

Модель базы данных находится во второй нормальной форме, если для каждого ее отношения выполняются следующие условия:

* отношение находится в первой нормальной формы;
* все поля записи отношения функционально полно зависят от первичного ключа.

В сущности**Диск** атрибуты **категория**, **жанр** зависят только от частей составного первичного ключа**код\_категории** и **код\_жанра** соответственно. Поэтому отношение «Диск» не находится во второй нормальной форме и его следует преобразовать, выделив из него отдельные сущности **Категория** и **Жанр**.

Приведем таблицу сущностей и атрибутов во второй нормальной форме (табл. 3).

Таблица 3.Атрибуты и первичные ключи сущностей во второй нормальной форме

| **Сущность** | **Первичный ключ** | **Атрибуты** |
| --- | --- | --- |
| Договор | код\_договора | код\_договора  код\_клиента  код\_сотрудника  дата\_проката  дата\_оконч\_дог  дата\_оконч\_факт  длительность\_дог  длительность\_факт  сумма  сумма\_за\_сутки  сумма\_залога |
| Жанр | код жанра | код\_жанра  код\_категории  жанр |
| Категория | код\_категории | код\_категории  категория |
| Состав |  | код\_договора  код\_диска  сумма\_за\_диск |
| Клиент | код\_клиента | код\_клиента  фам\_клиента  имя\_клиента  отч\_клиента  тел\_клиента  гор\_клиента  ул\_клиента  дом\_клиента  корп\_клиента  кв\_клиента  серия\_паспорта  номер\_паспорта |
| Сотрудник | код\_сотрудника | код\_сотрудника  фам\_сотрудника  имя\_сотрудника  отч\_сотрудника  тел\_сотрудника |
| Диск | код\_диска | код\_диска  код\_жанра  название  год\_выпуска  залог  цена\_сутки  наличие |

Приведем диаграмму взаимосвязей между атрибутами сущностей во второй нормальной форме (рис. 4).

|  |
| --- |
| **Диск** |
| код\_диска |
| название |
| код\_жанра |
| год\_выпуска |
| залог |
| цена\_сутки |
| наличие |
|  |
| **Жанр** |
| код\_жанра |
| жанр |
| код\_категории |
|  |
|  |
|  |
| **Категория** |
| код\_категории |
| категория |
|  |
|  |
|  |

|  |
| --- |
| **Договор** |
| код\_договора |
| код\_клиента |
| код\_сотрудника |
| дата\_проката |
| дата\_оконч\_дог |
| дата\_оконч\_факт |
| длительность\_дог |
| длительность\_факт |
| сумма |
| сумма\_за\_сутки |
| сумма\_залог |
|  |
|  |
|  |
|  |
| **Состав** |
| код\_договора |
| код\_диска |
| сумма\_за\_диск |

|  |
| --- |
| **Клиент** |
| код\_клиента |
| фам\_клиента |
| имя\_клиента |
| отч\_клиента |
| тел\_клиента |
| гор\_клиента |
| ул\_клиента |
| дом\_клиента |
| корпус\_клиента |
| кв\_клиента |
| серия\_паспорта |
| номер\_паспорта |
|  |
|  |
|  |
| **Сотрудник** |
| код\_сотрудника |
| фам\_сотрудника |
| имя\_сотрудника |
| отч\_сотрудника |
| тел\_сотрудника |

Рис.4. Диаграмма взаимосвязей между атрибутами сущностей во второй нормальной форме

### 2.2.6. Приведение модели базы данных к третьей нормальной форме

Модель базы данных находится во второй нормальной форме, если для каждого ее отношения выполняются следующие условия:

* отношение находится во второй нормальной формы;
* внутри каждогоотношения отсутствуют транзитивные зависимости.

В сущности**Диск** атрибут **цена\_сутки** зависит от атрибута **залог** этой же сущности и составляет 10% от величины залога (согласно описанию предметной области). Поэтому удалим данный атрибут из сущности, создав на форме поле, вычисляемое по формуле:

**Цена\_сутки=0.1\*залог**, (**1**)

где **дата\_оконч\_дог**, **дата\_проката** – поля сущности **Договор**.

В сущности**Договор** атрибут **длительность\_дог** зависит от атрибутов **дата\_проката** и **дата\_оконч\_дог** этой же сущности. Поэтому удалим данный атрибут из сущности, создав на форме поле, вычисляемое по формуле:

**Длительность\_проката=дата\_оконч\_дог–дата\_проката**, (**2**)

где **дата\_оконч\_дог**, **дата\_проката** – поля сущности **Договор**.

В сущности**Договор** атрибут **длительность\_факт** зависит от атрибутов **дата\_проката** и **дата\_оконч\_факт** этой же сущности. Поэтому удалим данный атрибут из сущности, создав на форме поле, вычисляемое по формуле:

**Длительность\_проката (факт)=дата\_оконч\_факт–дата\_проката**, (**3**)

где **дата\_оконч\_факт**, **дата\_проката** – поля сущности **Договор**.

В сущности**Договор** атрибут **сумма\_за\_сутки** зависит от атрибута **сумма\_залога** этой же сущности. Поэтому удалим данный атрибут из сущности, создав на форме поле, вычисляемое по формуле:

**Сумма\_за\_сутки=0.1\*Сумма\_залога**, (**4**)

где **дата\_оконч\_факт**, **дата\_проката** – поля сущности «Д**оговор**».

В сущности**Договор** атрибут «сумма» зависит от атрибутов **сумма\_за\_сутки**, **дата\_оконч\_дог**, **дата\_проката** и **сумма\_залога** этой же сущности. Поэтому удалим данный атрибут из сущности, создав на форме поле, вычисляемое по формуле:

**Сумма=сумма\_за\_сутки\*(дата\_оконч\_дог–дата\_проката)+сумма\_залога=**

**=0.1\*Сумма\_залога\*(дата\_оконч\_дог–дата\_проката)+сумма\_залога**, (**5**)

где**дата\_оконч\_дог**, **дата\_проката** и **сумма\_залога** – поля сущности **Договор**.

В сущности**Состав** отсутствует первичный ключ, и транзитивная зависимость вызвана наличием вычисляемого атрибута **сумма\_за\_диск**, поэтому удалим данное поле из сущности, создав на форме поле, вычисляемое по формуле:

**Сумма\_за\_диск=залог+цена\_сутки\*(дата\_оконч\_дог – дата\_проката)**, (**6**)

где **залог**, **цена\_сутки** – поля сущности **Диски**, **дата\_оконч\_дог**, **дата\_проката** – поля сущности **Договор**.

Приведем таблицу сущностей и атрибутов в соответствии с требованиями третьей нормальной формы (табл. 4).

Таблица 4. Атрибуты и первичные ключи сущностей в третьей нормальной форме

| **Сущность** | **Первичный ключ** | **Атрибуты** |
| --- | --- | --- |
| Договор | код\_договора | код\_договора  код\_клиента  код\_сотрудника  дата\_проката  дата\_оконч\_дог  дата\_оконч\_факт  сумма\_залога |
| Состав |  | код\_договора  код\_диска |
| Клиент | код\_клиента | код\_клиента  фам\_клиента  имя\_клиента  отч\_клиента  тел\_клиента  гор\_клиента  ул\_клиента  дом\_клиента  корп\_клиента  кв\_клиента  серия\_паспорта  номер\_паспорта |
| Сотрудник | код\_сотрудника | код\_сотрудника  фам\_сотрудника  имя\_сотрудника  отч\_сотрудника  тел\_сотрудника |
| Жанр | код жанра | код\_жанра  код\_категории  жанр |
| Диск | код\_диска | код\_диска  код\_жанра  название  год\_выпуска  залог  наличие |
| Категория | код\_категории | код\_категории  категория |

Приведем диаграмму взаимосвязей между атрибутами сущностей в третьей нормальной форме.

|  |
| --- |
| **Диск** |
| код\_диска |
| название |
| код\_жанра |
| год\_выпуска |
| залог |
| наличие |
|  |
|  |
| **Жанр** |
| код\_жанра |
| жанр |
| код\_категории |
|  |
|  |
|  |
| **Категория** |
| код\_категории |
| категория |
|  |
|  |
|  |

|  |
| --- |
| **Договор** |
| код\_договора |
| код\_клиента |
| код\_сотрудника |
| дата\_проката |
| дата\_оконч\_дог |
| дата\_оконч\_факт |
| сумма\_залог |
|  |
|  |
|  |
|  |
|  |
|  |
| **Состав** |
| код\_договора |
| код\_диска |

|  |
| --- |
| **Клиент** |
| код\_клиента |
| фам\_клиента |
| имя\_клиента |
| отч\_клиента |
| тел\_клиента |
| гор\_клиента |
| ул\_клиента |
| дом\_клиента |
| корпус\_клиента |
| кв\_клиента |
| серия\_паспорта |
| номер\_паспорта |
|  |
|  |
|  |
| **Сотрудник** |
| код\_сотрудника |
| фам\_сотрудника |
| имя\_сотрудника |
| отч\_сотрудника |
| тел\_сотрудника |

Рис.5. Диаграмма взаимосвязей между атрибутами сущностей в третьей нормальной форме

## 2.3.Физическое проектирование базы данных

На данном этапе используются понятия используемой для реализации системы управления базой данных (MSVisualFoxPro 9.0). В частности, речь идет об именах полей таблиц, типах данных и размерах полей.

### 2.3.1. Физическое описание модели базы данных

Для реализации автоматизированной системы учета дисков в DVD-прокате разработана база данных, состоящая из 7 таблиц.

Таблица 5. Таблица **Cостав** (sostav.dbf)

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя поля** | **Тип поля** | **Размер поля** | **Содержание** |
| kod\_pr | Integer | 4 | код\_договора |
| kod\_dis | Integer | 4 | код\_диска |

Таблица 6. Таблица **Сотрудник** (sotr.dbf)

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя поля** | **Тип поля** | **Размер поля** | **Содержание** |
| kod\_sotr | Integer(AutoInc) | 4 | код\_сотрудника |
| fam\_sotr | Character | 25 | фам\_сотрудника |
| im\_sotr | Character | 25 | имя\_сотрудника |
| ot\_sotr | Character | 25 | отч\_сотрудника |
| tel\_sotr | Character | 13[[1]](#footnote-2) | тел\_сотрудника |

Таблица 7. Таблица **Договор** (prokat.dbf)

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя поля** | **Тип поля** | **Размер поля** | **Содержание** |
| кod\_pr | Integer(AutoInc) | 4 | код\_договора |
| kod\_kl | Integer | 4 | код\_клиента |
| kod\_sotr | Integer | 4 | код\_сторудника |
| data\_pr | Date | 8 | дата\_проката |
| data\_dog | Date | 8 | дата\_оконч\_дог |
| data\_fakt | Date | 8 | дата\_оконч\_факт |
| summa\_zal | Numeric | 10.2 | сумма\_залог |

Таблица 8. Таблица **Клиент** (klient.dbf)

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя поля** | **Тип поля** | **Размер поля** | **Содержание** |
| kod\_kl | Integer(AutoInc) | 4 | код\_клиента |
| fam\_kl | Character | 25 | фам\_клиента |
| im\_kl | Character | 25 | имя\_клиента |
| ot\_kl | Character | 25 | отч\_клиента |
| tel\_kl | Character | 13 | тел\_клиента |
| g\_kl | Character | 25 | гор\_клиента |
| ul\_kl | Character | 25 | ул\_клиента |
| d\_kl | Character | 3 | дом\_клиента |
| k\_kl | Character | 3 | корпус\_клиента |
| kv\_kl | Character | 3 | кв\_клиента |
| ser | Integer | 4 | серия\_паспорта |
| nom | Integer | 6 | номер\_паспорта |

Таблица 9. Таблица **Жанр** (ganr.dbf)

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя поля** | **Тип поля** | **Размер поля** | **Содержание** |
| kod\_gan | Integer(AutoInc) | 4 | код\_жанра |
| ganr | Character | 25 | жанр |
| kod\_kat | Integer | 4 | код\_категории |

Таблица 10. Таблица **Диск** (disk.dbf)

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя поля** | **Тип поля** | **Размер поля** | **Содержание** |
| kod\_dis | Integer(AutoInc) | 4 | код\_диска |
| nazv | Character | 100 | название |
| kod\_gan | Integer | 4 | код\_жанра |
| god | Integer | 4 | год\_выпуска |
| zalog | Numeric | 10.0 | залог |
| v\_prok | Logiсal | 1 | находится\_в\_прокате |

Таблица 11. Таблица **Категория** (kategor.dbf)

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя поля** | **Тип поля** | **Размер поля** | **Содержание** |
| kod\_kat | Integer(AutoInc) | 4 | код\_категории |
| kateg | Character | 25 | категория |

Таблица 12. Первичные и внешние ключи таблиц базы данных

|  |  |  |
| --- | --- | --- |
| **Имя таблицы** | **Первичный ключ** | **Внешний ключ** |
| sotr.dbf | kod\_sotr |  |
| prokat.dbf | кod\_pr | kod\_kl  kod\_sotr |
| ganr.dbf | kod\_gan | kod\_kat |
| klient.dbf | kod\_kl |  |
| disk.dbf | kod\_dis | kod\_gan |
| kategor.dbf | kod\_kat |  |
| sostav.dbf |  | kod\_pr  kod\_dis |

### 2.3.2. Создание базы данных средствами MS Visual FoxPro 9.0

На данном этапе выполняется создание базы данных средствами выбранной СУБД.

Будущее приложение – это набор файлов:

* главный (стартовый) файл программы;
* базы данных;
* таблицы;
* формы;
* отчеты;
* меню;
* файлы графических изображений;
* программные модули и т.д.

Все компоненты (файлы), участвующие в приложении, объединены в проект.

**Проект** – это своеобразный контейнер для различных элементов будущего приложения. Сборкой всех файлов приложения в единый проект занимается Project Manager (Менеджер проекта). Это компонент, отражающий файлы проекта и имеющий средства для манипуляции этими файлами и проектом в целом. Менеджер проекта упрощает разработку приложения, т.к. все элементы приложения сгруппированы по разделам. Кроме того, запоминается расположение каждого включенного в проект элемента. Добавленные в проект файлы можно редактировать, удалять, запустить на выполнение. После завершения создания отдельных файлов проекта строится само приложение – .app или .exe.

Поскольку при разработке приложения будет создан именно набор файлов, то полезно определиться со структурой папок, предназначенных для хранения этих файлов.

Таблица 13. Размещение компонент проекта

|  |  |
| --- | --- |
| **Папка** | **Компонент проекта** |
| DVD | Проект |
| DVD\BD | База данных |
| DVD\Forms | Формы |
| DVD\Reports | Отчеты |
| DVD\Programs | Программы |

### ЗАДАНИЕ 1. Создание базы данных в проекте MSVisualFoxPro 9.0

1. Предположим, что приложение будет создаваться на диске D:. Создайте на диске D: структуру папок (рис.6) для размещения компонент проекта в соответствии с таблицей 13.*Запомните расположение объектов проекта* ***D:\DVD****.*

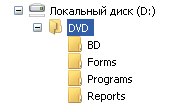


Рис. 6. Структура папок для размещения компонент проекта

1. Запустите MS Visual FoxPro 9.0.
2. Создайте в папке DVD проект с именем **dvd.pjx**:
   1. выполните команду меню **File – New**;
   2. в появляющемся окне запроса выберите переключатель **Project** и нажмите кнопку **Newfile**;
   3. в появляющемся окне выберите в качестве места расположения папку **D:\DVD**, введите имя проекта **dvd.pjx** и нажмите кнопку**Сохранить**.

В результате на экране отобразится окно **Project Manager**, в заголовке которого присутствует имя созданного проекта (рис. 7).

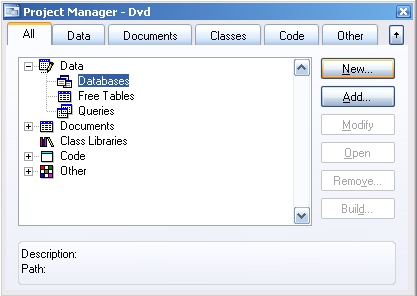


Рис. 7. Окно **Project Manager** после создания проекта

1. Создайте в проекте базу данных с именем **dvd-bd.dbc**:
   1. раскройте узел **Data**, выполнив щелчок на знаке «**+**»;
   2. выделите объект **Database** и нажмите кнопку **New**;
   3. в появляющемся окне запроса способа создания выберите создание с помощью конструктора – кнопка **New Database**;
   4. в появляющемся окне выберите в качестве места расположения папку **D:\DVD\BD**, введите имя базы данных **dvd-bd.dbc** и нажмите кнопку**Сохранить.**

В результате не экране отобразится окно схемы базы данных DatabaseDesigner, которое следует закрыть, т.к. пока база данных не содержит таблиц. Окно **ProjectManager** изменится: в узле **Database** отобразится созданная база данных (рис. 8).

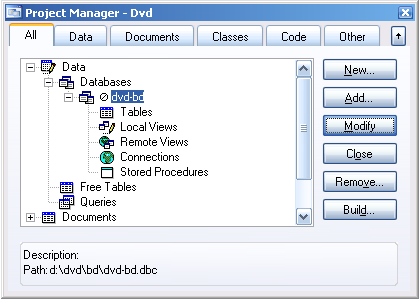


Рис. 8. Окно **ProjectManager** после создания в проекте базы данных

1. Создайте в базе данных таблицу **Sotr.dbf**:
   1. раскройте узел **Dvd-bd**, выполнив щелчок на знаке «+»;
   2. выделите объект **Tables** и нажмите кнопку **New**;
   3. в появляющемся окне запроса способа создания выберите создание с помощью конструктора – кнопка **New Table**;
   4. в появляющемся окне выберите в качестве места расположения папку D:\DVD\BD, введите имя таблицы базы данных **Sotr.dbf** и нажмите кнопку **Сохранить**;

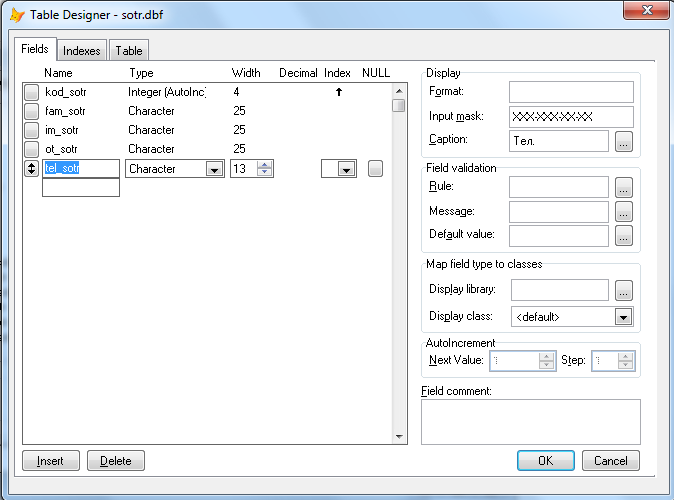


Рис. 9 – Окно конструктора таблицы **Table Designer**

* 1. в окне конструктора таблицы задайте поля, их типы, размеры согласно табл. 6; заполните в правой части поле **Caption** следующими данными: для поля **fam\_sotr** – Фамилия, **im\_sotr** – Имя, **ot\_sotr** – Отчество, **tel\_sotr** – Тел.; для поля **tel\_sotr** в правой части конструктора заполните поле **Input Mask** значением ххх-ххх-хх-хх (чтобы номер телефона отображался в формате ххх-ххх-хх-хх) (рис. 9);
  2. задайте первичный ключ: для поля **kod\_sotr**измените значение столбца **Index**, выбрав значение ↑**Ascending** или ↓**Descending** (рис. 10) – поле станет ключевым; перейдите на вкладку **Indexes** и в столбце **Type** выберите значение **Primary** – ключевое поле станет полем первичного ключа;

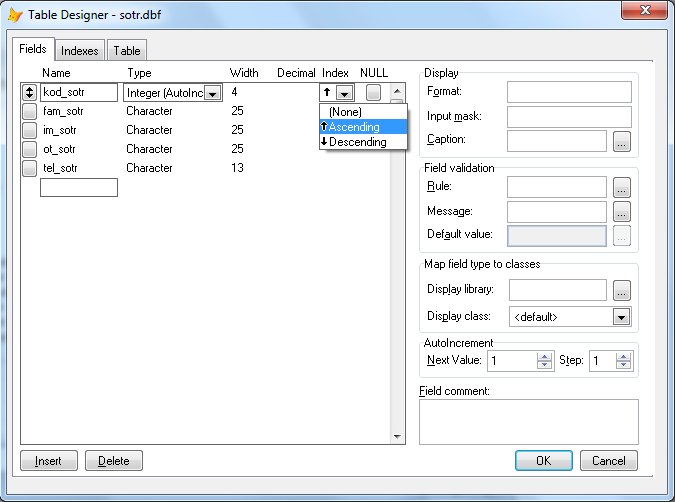


Рис. 10 – Окно конструктора таблицы **Table Designer**

* 1. нажмите кнопку **ОК**.

Обратите внимание: в окне проекта объект **Tables** стал узловым, раскройте этот узел и убедитесь, что в нем отображается созданная таблица.

1. Раскройте узел созданной таблицы и обратите внимание на его содержимое: отображаются поля таблицы и особым образом отмечено поле первичного ключа (рис. 11).

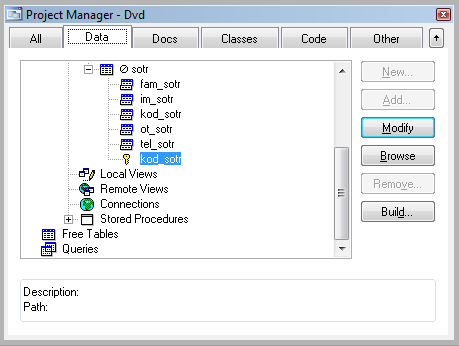


Рис. 11 – Структура таблицы **Sotr**в окне **Project Manager**

1. Повторяя действия пункта 5, создайте в проекте таблицу **Disk.dbf**. Ее структура описана в табл.10. Установите значение свойства**Caption** для полей: **god** – Год, **v\_prok** – Выдан, **nazv** – Название, **zalog** – Залог. Задайте первичный ключ – **kod\_dis**. Согласно табл. 12кроме первичного ключа в этом отношении присутствует и внешний ключ **kod\_gan**. Чтобы его задать, сделайте поле **kod\_gan** ключевым (ЗАДАНИЕ 1, п.5-е) и на вкладке **Indexes** в столбце **Type** выберите значение **Regular** – ключевое поле станет полем внешнего ключа.
2. Раскройте узел таблицы **Disk.dbf** и убедитесь, что его содержимое совпадает с представленным на рис. 12. Обратите внимание, что особым образом отмечен внешний ключ таблицы.

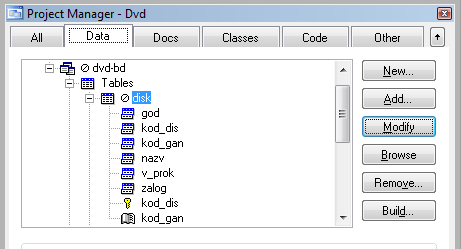


Рис. 12 – Структура таблицы **Disk** в окне **ProjectManager**

1. Повторяя действия пункта 5, создайте в проекте таблицу **ganr.dbf**. Ее структура описана в табл.9. Установите значение свойства **Caption** для поля: **ganr** – Жанр. Задайте первичный ключ в соответствии с табл. 12.
2. Повторяя действия пункта 5, создайте в проекте таблицу **prokat.dbf**. Ее структура описана в табл. 7. Установите значение свойства **Caption** для полей: **data\_dog** – Дата окончания по договору, **data\_fakt** – Дата возврата фактическая. Задайте первичный и внешние ключи в соответствии с табл. 12.
3. Повторяя действия пункта 5, создайте в проекте таблицу **sostav.dbf**. Ее структура описана в таблице 3. Задайте внешние ключи в соответствии с табл.12.
4. Повторяя действия пункта 5, создайте в проекте таблицу **kategor.dbf**. Ее структура описана в табл. 11. Установите значение свойства **Caption** для поля **kateg** – Категория. Задайте первичный ключ в соответствии с табл.12.
5. Повторяя действия пункта 5, создайте в проекте таблицу **klient.dbf**. Ее структура описана в табл.8. Установите значение свойства **Caption** для полей: **kod\_kl** – №, **fam\_kl** – Фамилия, **im\_kl** – Имя, **ot\_kl** – Отчество, **tel\_kl** – Телефон, **g\_kl** – Город, **ul\_kl** – Улица, **d\_kl**– Дом, **k\_kl** – Корпус (строение), **kv\_kl** – Квартира, **ser** - Паспорт. Задайте первичный ключ в соответствии с табл.12.
6. В окне проекта выделите созданную базу данных **dvd-bd.dbc** и нажмите кнопку **Modify** для отображения схемы базы данных **DatabaseDesigner** (рис. 13).Убедитесь, что в окне отображается 7 таблиц базы данных, каждая таблица содержит необходимы поля и установленные первичные и внешние ключи (обратите внимание, что ключевые поля обозначены отдельно в области **Indexes** каждой таблицы). При необходимости отредактируйте каждую таблицы, используя команду **Modify** контекстного меню таблицы в окне схемы данных или кнопку **Modify** в окне менеджера проекта*. Расположение и размеры таблиц в окне* ***DatabaseDesigner*** *могут отличаться от представленного, при необходимости их можно изменить протяжкой или перетаскиванием мышью. Если связь установлена неверно, то ее можно удалить, выделив и нажав клавишу* ***Delete***

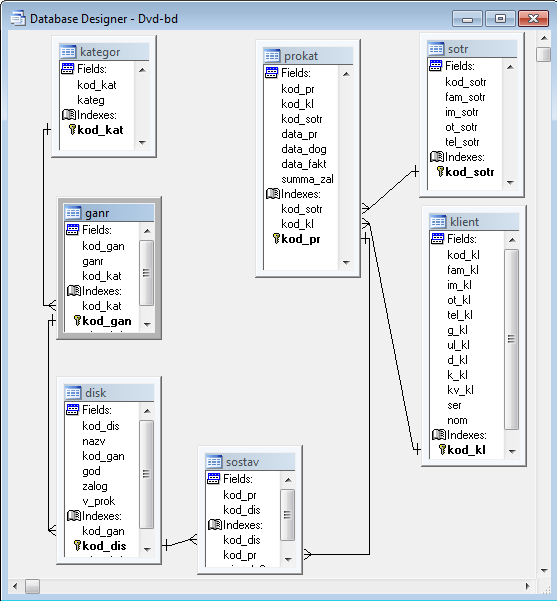


Рис. 13 – Окно схемы базы данных

1. Убедившись, что таблицы базы данных созданы верно, создайте связи между таблицами в соответствии с диаграммой, представленной на рис. 5:
   1. для создания связи, например, между таблицами **Ganr** и **Kategor** выделите первичный ключ **kod\_kat** в таблице **Kategor** в области **Indexes** и перетащите его на соответствующий внешний ключ в области **Indexes** таблицы **Ganr**;
   2. создайте связи между остальными таблицами.
2. Убедитесь, что схема базы данных соответствует представленной на рис. 13.
3. Завершите работу с проектом.

# 3. ЗАПОЛЕНИЕ И ПРОСМОТР ТАБЛИЦ БАЗЫ ДАННЫХ В РЕЖИМЕ BROWSE

## 3.1. Заполнение и просмотр таблиц

Итак, база данных создана. Теперь следует заполнить ее таблицы информацией. Заполнение таблиц может быть выполнено в двух режимах:

* с помощью форм;
* в режиме таблицы.

Заполнение таблиц с помощью форм – вариант, наиболее удобный для потенциального пользователя будущего приложения. Данный вариант позволяет реализовать наиболее удобный интерфейс, который не требует знания приемов работы в FoxPro.

Использование режима таблицы для ввода данных – менее приемлемый с точки зрения потенциального пользователя вариант, т. к. требует знания специальных команд.

Но для выполнения следующих заданий пособия необходимо наличие таблиц с данными. Поэтому часть таблиц будут заполнены именно в этом режиме. Хотя некоторые таблицы в этом режиме заполнять вообще не рекомендуется.

Не существует какого-то строгого правила относительно того, какие таблицы лучше заполнять с помощью форм. Но все таблицы в разрабатываемой нами базе данных можно условно поделить на три вида:

1. **Kategor**, **Klient**, **Sotr** – они не содержат внешних ключей;
2. **Ganr**, **Disk** – в них есть внешние ключи;
3. **Prokat** – главная таблица, позволяющая вести учет дисков.

Таблица **Sostav** не рассматривается – это промежуточная таблица, не имеющая первичных ключей.

В дальнейшем заполнение таблиц базы данных лучше и вести именно в таком порядке.

В системе Visual FoxPro существует два основных режима работы с таблицами (и, соответственно, команды языка этой системы): **Browse** - табличный формат и **Edit** - форма с построчным расположением полей.

Существует много разных способов открыть окно таблицы базы данных для редактирования в одном из этих режимов, в том числе:

* в открытом окне модификации структуры базы (**DatabaseDesigner**) щелкнуть правой кнопкой мыши на таблице и выбрать команду **Browse** ;
* выбрать нужную таблицу в менеджере проекта и нажать на кнопку **Browse** .

После выполнения этих действий таблица БД будет открыта для редактирования данных, а в пункте меню появляются команды выбора режима **Browse** или **Edit**.

Для добавления данных в режиме таблицы следует использовать команду меню **Table – Append New Record**иликомбинациюклавиш**Ctrl+Y**.

### ЗАДАНИЕ 2. Заполнение и просмотр таблиц базы данных

1. Запустите проект **dvd.pjx**.
2. Откройте для просмотра и заполнения таблицу **Kategor.dbf**:
   1. выделите в менеджере проекта таблицу **Kategor.dbf** и нажмите кнопку **Browse**;
   2. нажмите комбинацию клавиш Ctrl+Y для добавления первой записи;
   3. обратите внимание, что столбец **kod\_kat** заполняется автоматически, т.к. имеет тип данных **Integer (AutoInc)**;
   4. используя для перехода по ячейкам таблицы клавиши управления курсором или Tab, введите первую категорию – Кино;
   5. действуя аналогично, добавьте в таблицу еще две записи, как показано на рис. 14 (не забывайте для добавления каждой записи нажимать комбинацию клавиш **Ctrl+Y**). Закройте таблицу после заполнения.

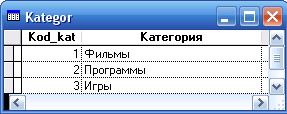


Рис. 14 –Заполненная таблица **Kategor.dbf**

Обратите внимание на разницу в написании имен столбцов таблицы при просмотре. Первый столбец **kod\_kat** назван так же, как и соответствующее поле в таблице базы данных. Второй столбец имеет имя не поля **Kategor**, а его свойства **Caption** – **Категория**.

Размеры столбцов и высоту строк таблиц можно менять аналогично подобным действиям с таблицами MSExcel.

1. Повторяя действие п. 2, заполните таблицу **Klient.dbf** следующими данными (рис. 15):

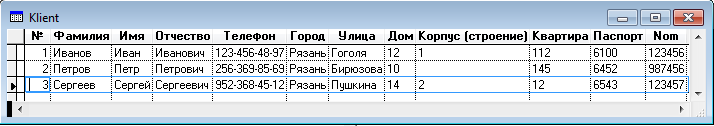


Рис. 15 – Заполненная таблица **Klient.dbf**

1. Повторяя действия п. 2, заполните таблицу **Sotr.dbf** следующими данными (рис. 16):

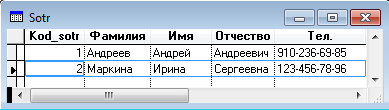


Рис. 16 – Заполненная таблица **Sotr.dbf**

1. Заполните таблицу **Ganr.dbf** 10-ю записями. Необходимо ввести в таблицу 4 жанра категории «**Фильм**»: Фантастика, Мистика, Ужасы, Комедия; 3 жанра категории «**Программы**»: Графика, Мультимедиа, Драйвера; 3 жанра категории «**Игра**»: Шутер-Action, Стратегия, Гонки. Действуйте аналогично п.2, а чтобы отнести жанр к определенной категории в поле **kod\_kat** вводите код категории в соответствии с таблицей **Kategor** (рис. 17).

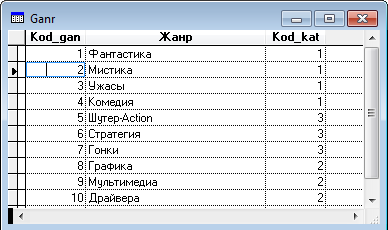


Рис. 17 – Заполненная таблица **Ganr.dbf**

1. Заполните таблицу **Disk.dbf** следующими данными:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **kod\_dis** | **Название** | **Жанр** | **Год** | **Залог** |
|  | Колония | Фантастика | 2013 | 200 |
|  | Я, Франкенштейн | Ужасы | 2014 | 300 |
|  | Особое мнение | Фантастика | 2002 | 250 |
|  | Этот неловкий момент | Комедия | 2014 | 280 |
|  | Тихоокеанский рубеж | Фантастика | 2013 | 270 |
|  | Metro: Last Light | Шутер-Action | 2013 | 350 |
|  | **Game Stock Car** | Гонки | 2013 | 350 |
|  | Akvis All Plugins | Графика | 2014 | 250 |
|  | Test Drive Unlimited 2 | Гонки | 2011 | 300 |
|  | ZBrush Studio | Графика | 2013 | 280 |
|  | DriverPack Solution | Драйвера | 2014 | 280 |
|  | Wargame: Red Dragon | Стратегия | 2014 | 300 |
|  | Pinnacle Studio | Мультимедиа | 2013 | 350 |
|  | В тылу врага: Штурм 2 | Стратегия | 2014 | 300 |
|  | Огневой рубеж | Комедия | 2013 | 260 |

Для отнесения диска к определенному жанру используйте столбец **kod\_gan** (как при отнесении жанра к определённой категории в п. 5).

1. Завершите работу с проектом.

## 3.2. Просмотр и отбор содержимого таблиц с помощью команды Browse

Команды управления базой данных вводятся в системном окне **Command** (рис. 18). Если оно не отображается в окне программы, то для его вывода можно использовать кнопку **CommandWindow** панели инструментов.

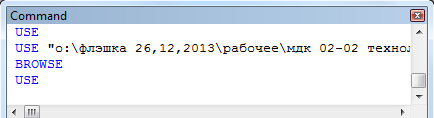


Рис. 18 – Системное окно **Command**

Окно **Command** позволяет выполнять следующие действия:

* Вводить команды и запускать их на выполнение нажатием клавиши **Enter**.
* Выполнять блок кода прямо в командном окне, предварительно выделив его и выбрав команду контекстного меню **Execute Selection**.
* Удалить команду, если еще не нажата клавиша **Enter**.
* Вводить длинные команды, разделяя строки точкой с запятой.
* Переносить текст команды из окна **Command** в другое редактируемое окно.
* Повторно выполнять команды, поместив курсор на строку с командой и нажав **Enter**.

Команда **Browse** используется для отображения содержимого текущей или выбранной таблицы. При разработке приложений команда используется для написания процедур. Вообще говоря, команда имеет большое количество параметров, но рассмотрим только те из них, которые будут использоваться практически при выполнении заданий:

**BROWSE [FIELDS список\_полей] [FOR Условие] [TITLE Текст\_заголовка]**

Список\_полей – перечисление полей через запятую, если нужно отобразить не все поля, а только часть из них.

Условие – показываются только записи, которые удовлетворяют условию. Можно использовать функции и операции сравнения;

Текст\_заголовка – заменяет помещаемое в заголовок окна имя таблицы на заданный вами текст;

Список\_полей может включать любую комбинацию полей базы данных и/или вычисляемых полей. Формат списка следующий:

**<поле1> [:R] [:nn][:H = <заголовок>]<имя вычисляемого поля>=<выражение>**

:nn определяет ширину выводимой колонки и является необязательным;

:Н= определяет заголовок для поля при выводе его на экран, удобно для вычисляемых полей.

:R – поле «только для чтения»

Примеры:

1. Вывод только тех записей таблицы, в которых значение поля «cena» меньше 100:

**BROWSE FOR** cena<100

1. Вывод не всех столбцов таблицы, а только Kod\_tovara (с присвоением заголовка «Номер» и установкой ширины 10), Tovar (с присвоением заголовка «Товар» и установкой ширины 50), причем при выводе таблице присваивается заголовок «Список товаров»:

**BROWSE Fields** Kod\_tovara:H=’Номер’:10, Tovar:H=’Товар’:50 Title ‘Список товаров’

1. Вывод не всех столбцов таблицы, а только Kod\_tovara (с присвоением заголовка «Номер» и установкой ширины 10), Tovar (с присвоением заголовка «Товар» и установкой ширины 50) и вычислимого поля Nacenka:

**BROWSE Fields** Kod\_tovara:H=’Номер’:10, Tovar:H=’Товар’:50, Nacenka=cena\*0.2

### ЗАДАНИЕ 3. Отображение и отбор содержимого таблиц

1. Запустите проект **dvd.pjx**.
2. В свободной рабочей области откройте таблицу ganr.dbf: введите в окне **Command** команду **USED:\DVD\BD\ganr.dbf**.
3. Просмотрите содержимое таблицы Ganr.dbf, введя в окне **Command** команду **Browse**.
4. Закройте таблицу в текущей рабочей области, введя команду **USE**.
5. Аналогично п.2, откройте в свободной рабочей области таблицу Disk.dbf.
6. Выведите сведения о дисках, выпущенных в 2014 году. Для этого введите команду **Browseforgod=2014**. *Обратите внимание, что при выводе отображаются все поля таблицы*. Закройте таблицу стандартным способом закрытия окон.
7. Выведите название и залог дисков, если залог больше 300. Для этого введите команду **Browsefieldsnazv, zalogforzalog>300**. *Обратите внимание, что отображаются только два поля таблицы*.
8. Выведите название и залог дисков, если залог больше 300, но меньше 380, введя команду **Browsefieldsnazv, zalogforzalog>300 andzalog<380.**
9. Выведите сведения о фантастических фильмах 2013 года. Результатом должна стать таблица «Фантастика 2013» с заголовками столбцов: Диск, Жанр\_диска, Год. Для этого введите команду**Browsefieldsnazv:H=’Диск’, kod\_gan:H=’Жанр\_диска’, god:H=’Год’ forgod=2013 andkod\_gan=1 TITLE ‘Фантастика 2013’**. *Обратите внимание, что отображаются новые названия для полей таблицы и новый заголовок самой таблицы*.
10. Выведите сведения о диске с названием «Колония». Для этого введите команду **Browsefornazv=’Колония’**.
11. Выведите сведения о фильмах, в названии которых есть слово «рубеж», задав заголовок таблицы результатов. Для этого введите команду **Browseforatcc(’рубеж’, nazv)<>0TITLE ‘Содержат слово’**.[[2]](#footnote-3)
12. Выведите сведения о дисках, причем отображаются названия дисков, стоимость залога и за\_сутки, а также сколько процентов от залога составляет стоимость за сутки. Для этого введите команду **Browsefieldsnazv, zalog, cena=zalog\*0.1:H=’Цена\_сут’.** *Обратите внимание, что отображается столбец, отсутствующий в таблице базы данных*. Закройте таблицу просмотра.
13. Выделите в проекте таблицу disk.dbf и нажмите кнопку Browse. Обратите внимание, что таблица содержит заголовок и столбцы, настроенные при предыдущей команде просмотра. Чтобы вернуть исходный вид таблице просмотра, введите команду **Browse** без параметров.
14. Закройте таблицу командой **USE**.
15. Завершите работу спроектом.

## 3.3. Поиск данных в таблице

Вообще говоря, MSVisualFoxPro предлагает несколько команд, связанных с поиском данных. Далее рассмотрены те из них, которые будут использованы при выполнении практических заданий.

Команда **Locate** используется для поиска в таблице базы данных записей, удовлетворяющих определенному условию.

**LOCATE [FOR <Условие1>] [Диапазон] [WHILE <Условие2>]**

Условие1 – условие поиска; ищется первая запись, удовлетворяющая этому условию;

Диапазон – задает диапазон поиска и может принимать значения:

ALL -все записи,

NEXT <Количество> – указанное количество следующих за текущей записей,

RECORD <Номер\_записи> – поиск в конкретной записи,

REST – поиск начиная с текущей записи и до конца таблицы;

Условие2 – задает выражение, при истинности которого продолжается поиск;

Команда LOCATE ищет запись, соответствующую условиям поиска, и делает ее текущей. Продолжить поиск после нахождения первой записи, удовлетворяющей условиям поиска, можно посредством команды CONTINUE. Поиск в этом случае будет продолжен с записи, следующей за текущей.

Определить, найдена ли запись командой **LOCATE**, можно, если после их выполнения оценить значение, возвращаемое функцией **FOUND()**.

**FOUND([<Номер\_рабочей\_области> | <Алиас\_таблицы>])**

<Номер\_рабочей\_области> | <Алиас\_таблицы> - обозначение таблицы, в которой производится поиск.

Если функция **FOUND()** вернула значение **True** (**.Т.**), значит, искомое значение найдено, если функция **FOUND()**вернула значение **False** (.**F.**) – искомое значение не найдено.

После осуществления поиска рекомендуется выводить на экран сообщения пользователю, если поиск не дал результатов. Для формирования диалоговых окон сообщений используется функция **messagebox()**.

**MESSAGEBOX(eMessageText [, nDialogBoxType ][, cTitleBarText][, nTimeout])**

eMessageText – текстсообщения;

nDialogBoxType – типокна:

0 – только кнопка ОК

1 – кнопки ОК и Отмена

2 – кнопки Прервать, Повтор, Пропустить

3 – кнопки Да, Нет, Отмена

4 – кнопки Да, Нет

5 – кнопки Повтор, Отмена

16 – Стоп-знак

32 – Вопрос-знак

48 – Восклицательный знак

64 – i-знак

cTitleBarText – заголовок окна;

nTimeout – задает количество миллисекунд, в течение которых Visual FoxPro отображает eMessageText без ввода с клавиатуры или мыши перед очисткой eMessageText. Вы можете указать любое допустимое значение тайм-аута. Если значение меньше 1, то тайм-аут никогда не истекает до тех пор, пока пользователь вводит входные. То же самое происходит, когда параметр опущен.

Возвращаемые значения: 1 – OK, 2 – Отмена, 3 – Прервать, 4 – Повтор, 5 – Пропустить, 6 – Да, 7 – Нет.

### ЗАДАНИЕ 4. Поиск данных в таблице

1. Запустите проект **dvd.pjx**.
2. В окне Command выполните команду **SETDEFAULTTOD:\DVD**. Эта команда позволит установить в качестве рабочего каталога папку D:\DVD – папку, в которой хранятся все компоненты проекта. В дальнейшем при обращении к объектам данную часть пути можно не указывать.
3. Откройте в свободной рабочей области таблицу Disk.dbf, введя команду **USEbd\disk.dbf**. *Обратите внимание, что в команде не указана часть пути D:\DVD.*
4. Выполните поиск в таблице Disk дисков, выпущенных в 2014 году. Для этого введите команду **LOCATEFORgod=2014**. *Обратите внимание, что после запуска команды никакой результат не отображен.*
5. Чтобы увидеть результат работы команды, необходимо после поиска (LOCATE) выполнить вывод (BROWSE) найденных записей, **если** такие **записи найдены**(FOUND). При этом требуется условная конструкция **IF**, а саму процедуру поиска лучше оформить в отдельной программе.
   1. в окне менеджера проекта перейдите на вкладку **Code**, выделите объект **Program** и нажмите кнопку **New**;
   2. в окне ввода кода введите текст процедуры:

**LOCATE FOR** god=2014

**IF FOUND()**=.t. **THEN**

**BROWSE FOR** god=2014 **TITLE** 'Диски 2014 года'

**ENDIF**

* 1. закройте окно редактора кода, при запросе выбрав в качестве места сохранения папку D:\DVD\Programs и имя программы **Disk-2014.prg**;
  2. проверьте правильность работы программы, выделив ее в окне проекта и нажав кнопку **Run**;
  3. закройте окно результатов.

1. Откройте созданную программу для редактирования, выделив ее в менеджере проектов и нажав кнопку **Modify**. Скопируйте текст программы привычным для текстовых редакторов способом.
2. Создайте в менеджере проектов еще одну программу (п.5). Измените в первой и третьей строках год на «2015». Сохраните ее под именем **Disk-2014.prg**и запустите. Обратите внимание, что в результате отображается окно просмотра таблицы с пустыми записями. Закройте окно результатов.
3. Отредактируйте программу таким образом, чтобы при поиске, не давшем результатов, на экране появлялось соответствующее текстовое сообщение. Откройте программу для редактирования (п.6) и измените ее содержимое:

**LOCATE FOR** god=2015

**IF FOUND()**=.t. **THEN**

**BROWSE FOR** god=2015 **TITLE** 'Диски 2014 года'

**ELSE**

**MESSAGEBOX(**'Диски не найдены'**,**16**,**'Сообщение'**)**

**ENDIF**

1. Завершите работу с проектом.

# 4. РАЗРАБОТКА ФОРМ ДЛЯ РАБОТЫ С БАЗОЙ ДАННЫХ

Приложения Windows, использующие графический интерфейс пользователя, для отображения информации используют окна. Окном называется прямоугольная область экрана, связанная с приложением. События (**Events**), такие как перемещение или изменение размеров окна, щелчок ЛКМ на области экрана, занимаемой окном, трансформируются в сообщения, посылаемые Windows этому окну. Если окно активно, то ему также направляются сообщения о событиях, происходящих при нажатии клавиш на клавиатуре.

В Visual FoxPro окна создаются из базового класса FORM и называются формами.

**Форма** – объект, предназначенный для ввода данных, отображения их на экране или управления работой приложения. Объекты FORM, представляющие документы, могут выглядеть на экране, как бланки этих документов. Форма – это основной инструмент разработки пользовательского интерфейса приложения.

Формы как объекты предоставляют разработчику множество свойств, позволяющих управлять внешним видом окна, например, шириной (Width), высотой (Height) и т. п. Кроме того, формы включают в себя методы (обработчики некоторых сообщений Windows), например, сообщения о событиях Click, DblClick и RightClick, вызванных щелчком мыши на форме. Объект FORM предоставляет так же ряд методов, непосредственно связанных с окном и расширяющих его собственную функциональность, например, методы Load, Release. Но самое важное – это то, чтоб объект FORM является контейнером, позволяющим на созданном окне размещать различные элементы управления, что позволяет реализовать интерактивный диалог с пользователем. Это и делает форму основным компонентом приложения. Объектами формы могут являться поле, надпись, раскрывающийся список, кнопка и т.д.

**Способы создания форм**

1. Мастер (**Wizard**). Автоматическое создание формы.
   1. **FormWizard** – создание формы на основе одной таблицы.
   2. **One-to-ManyFormWizard** – создание формы на основе 2-х связанных таблиц (связь 1:N)
2. Конструктор (**New**). Создание формы и ее компонент полностью вручную.

**Способы запуска формы**

1. Кнопка **Run** менеджера проекта.
2. Ввод команды **DOFORM** имя\_формы.

**Способы закрытия формы**

1. Кнопка **EXITFORM** на панели кнопок.
2. Стандартное закрытие окна Windows.
3. Ввод команды **RELEASE()**.

**Источники данных для формы**

Обычно форма служит именно для работы с записями базы данных: просмотр, редактирование, добавление, удаление, изменение значений. В таких случаях форма привязана к определенным таблицам базы данных. Эти таблицы можно просмотреть или добавить новые источники данных с помощью окна **Data Environment**, вызываемого с помощью одноименной команды контекстного меню свободной части формы.

Существуют формы, не имеющие источников данных. Примером таких форм могут быть формы авторизации пользователя.

**Элементы управления формы**

Для создания новых элементов в форме используется панель инструментов **Form Controls Toolbar**, на которой размещены соответствующие кнопки. Панель инструментов отображается в режиме конструктора формы. Если панель скрыта, то ее можно отобразить, выполнив команду меню **View** - **Form Controls Toolbar**

Настройка элементов управления может выполняться с помощью окна **Properties**, некоторые элементы имеют встроенные построители, вызываемые с помощью команды **Builder** контекстного меню объекта.

Для каждого элемента формы и для самого окна формы существует определенный набор свойств.

## 4.1. Создание простейших форм

Вообще говоря, не существует точных признаков, которые позволяют какие-то формы отнести к простейшим, какие-то – к более сложным, какие-то – к самым сложным. В рамках данного пособия к простейшим формам буду отнесены формы, не содержащие вычислений и других компонент, кроме надписей и текстовых полей.

**Элемент Label (надпись)**

Служит для вывода на форме статического текста, заголовков, и поясняющих надписей. Многим приходилось заполнять бланки и анкеты. Они построены по одному принципу: есть обозначение какого-то пункта и есть пустое место для того, чтобы что-то написать. Label – это и есть «обозначение какого-то пункта».

Объект представляет собой текст, заключенную в некоторую ограничивающую рамку (представлена 6-ю маркерами):



Рис. 19 – Надпись в конструкторе формы

**Некоторые свойства Label**

**Caption** содержание надписи

**FontCharSet** шрифт, начертание и размер

**FontUnderline** подчеркивание

**FontBold** полужирное начертание

**FontSize** размер шрифта

**FontItalic** курсивное начертание

**Alignment** выравнивание внутри рамки надписи

**FontName** шрифт

**ForeColor** цвет текста внутри надписи

**Visible** видимость надписи

**Элемент TextBox (текстовое поле)**

Текстовое поле, в которое можно вводить информацию – числа, даты или текст – неотъемлемая часть многих форм.

TextBox

Label



Рис. 20 – Текстовое поле

Данные для текстового поля могут браться из таблицы базы данных или наоборот – заноситься в таблицу базы данных. Если же вы не хотите связывать **TextBox** с таблицей, то можно связать поле с переменной или не связывать ни с чем, а просто использовать для вычислений на форме. Данное в **TextBox** принадлежит к определенному типу данных. Это следует учитывать при вычислениях.

Для работы с **Text Box** существует встроенный построитель, вызываемый командой **Builder** из контекстного меню Text Box. Построитель позволяет:

* Изменить формат отображаемого значения.
* Изменить стиль поля.
* Изменить источник данных для поля.

**Некоторые свойства TextBox**

**ControlSource** Поле таблицы БД, которое будет связано с TextBox

**Visible** Видимость поля на форме

**Name** Имя поля на форме

**Value** Значение, содержащееся в поле

**Refresh** Обновление значения поля.

### ЗАДАНИЕ 5. Создание простейших форм ввода данных

1. Запустите проект **dvd.pjx**.
2. С помощью мастера форм создайте форму для просмотра и добавления записей в таблицу **Categor.dbf**. Для этого перейдите на вкладку **Documents (Docs)**, выделите объект **Forms** и нажмите кнопку **New**. В появляющемся окне запроса способа создания выберите **FormWizard**. В результате будет запущен мастер форм. Выберите тип мастера – **Form Wizard**.
3. Выполните шаги мастера:
   1. Выбор полей: выбранные здесь поля будут отображены на форме и позволят вносить в таблицу соответствующие данные. Используя кнопки  или , добавьте в область **SelectedFields** оба поля: **kod\_kat** и **Kategor** (рис. 21).

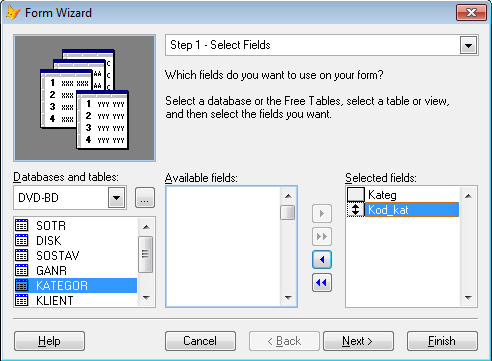


Рис. 21 – Мастер форм – шаг 1

* 1. Выбор стиля формы: внешний вид формы, кнопки перехода по записям (с надписями или с пиктограммами) (рис. 22);

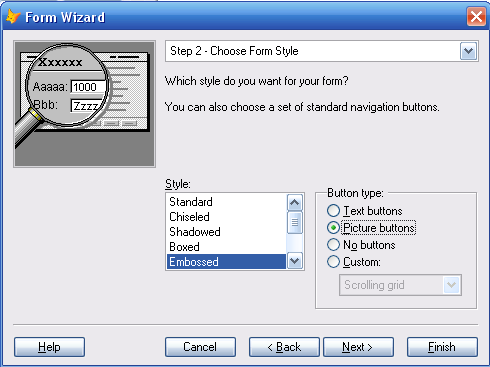


Рис. 22 – Мастер форм – шаг 2

* 1. Сортировка записей: можно указать поле, с сортировкой по которому записи будут отображаться в полях формы, а также порядок сортировки – по возрастанию (**Ascending**) или убыванию (**Descending**). Обычно, сортировка не выбирается и автоматически ведется по полю первичного ключа (рис. 23).

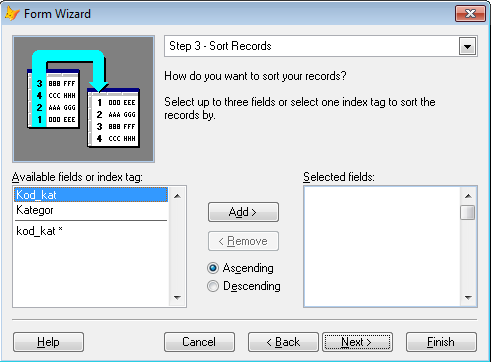


Рис. 23 – Мастер форм – шаг 3

* 1. Завершение (рис. 24):
     1. Заголовок формы: заголовок будущего диалогового окна. По умолчанию совпадает с названием таблицы, для которой создается форма.
     2. Варианты сохранения (сохранить и запустить, просто сохранить, сохранить и открыть конструктор).
     3. Предпросмотрформы.

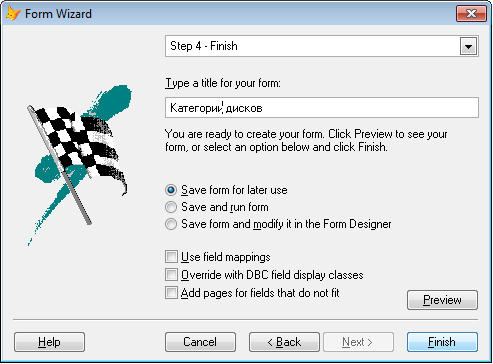


Рис. 24 – Мастер форм – шаг 4

* 1. Сохраните созданную форму под именем **Kategor** в папке **D:\DVD\Forms**. *Обратите внимание, что в папке появились два фала:* ***Kategor.scx*** *и* ***Kategor.sct****.*

1. Обратите внимание, что в менеджере проекта в группе **Forms** появилась созданная форма (рис. 25):

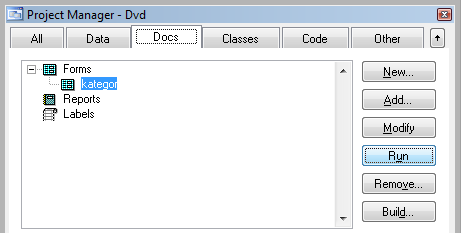


Рис. 25 – Менеджер проекта после создания формы

1. Запустите форму, выделив ее в окне менеджера проекта и нажав кнопку **Run**(рис. 26).

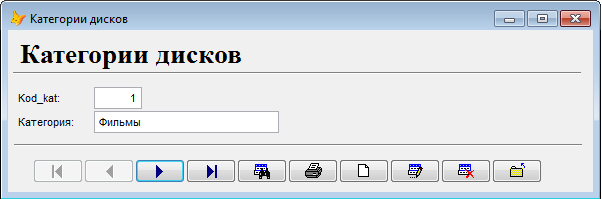


Рис. 26 – Запущенная форма Kategor.scx

Она содержит 3 вида элементов:

* надписи («Категории дисков», «Kod\_kat::»,«Категория:»);
* поля (в которых отображается содержимое полей таблицы kod\_kategor, kategor);
* панель кнопок:

 – Toprecord – переход к первой записи таблицы базы данных;

 – Previousrecord – переход к предыдущей записи таблицы базы данных;

 – Nextrecord – переход к следующей записи таблицы базы данных;

 – Bottomrecord – переход к последней записи таблицы базы данных;

 – Findrecords – поиск записей в таблице базы данных;

 – Printreport – вывод и печать отчета;

 – Addrecord – добавление записи в таблицу базы данных;

 – Saverecord – сохранение запись в таблице базы данных (кнопка появляется на месте кнопки для добавления записи);

 – Editrecord – редактирование текущей записи формы;

 – Revertrecord – отмена изменения в текущей записи таблицы базы данных;

 – Deleterecord – удаление текущей записи таблицы базы данных;

 – Exit form – закрытие формы.

1. Попробуйте первые четыре кнопки – перемещение по записям. *Обратите внимание, что в каждом экране формы отображается одна запись (строка) таблицы Kategor.dbf*.
2. Добавьте новую категорию «Музыка» в таблицу Kategor.dbf. Для этого нажмите на форме кнопку **Addrecord**, в результате чего поля формы станут доступными для добавления записей (рис. 27). Обратите внимание, что поле kod\_kat заполняется автоматически из-за типа данных этого поля – Integer(AutoInc). В поле Категория введите назавание категории Музыка и нажмите кнопку **Saverecord**. Завершите работу с формой, нажав кнопку **Exit form**.

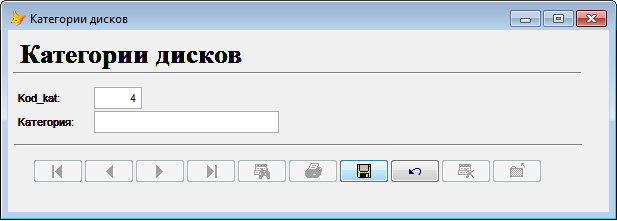


Рис. 27 – Форма при добавлении новой записи

1. Повторяя действия п. 2 и п. 3, создайте с помощью мастера формы «Клиенты», «Сотрудники», «Жанры» для просмотра и редактирования таблиц Klient,Sotrudnik и Ganr соответственно.
2. Добавьте в таблицу Klient запись еще об одном клиенте с помощью созданной формы.
3. Завершите работу с проектом.

## 

## 4.2. Применение конструктора форм

Для работы с формами, кроме мастера, можно использовать конструктор форм. Используя конструктор, можно создать форму «с нуля», но удобнее использовать конструктор не для создания, а для редактирования формы, уже созданной с помощью мастера. Именно в режиме конструктора доступно изменение свойств самой формы и ее объектов.

Для работы с объектами формы используется панель свойств **Properties** (рис. 28), которая показывает свойства, события, методы и их значения для выбранного объекта. Чтобы отобразить панель свойств, следует выполнить команду **Properties** контекстного меню любой части формы.

Если свойства, события и методы недоступны для редактирования, их значения показаны в списке курсивом. В верхней части окна показывается выбранный объект и при открытии списка отображается перечень доступных объектов. Список включает форму или набор форм, а также все включенные в форму объекты.

Свойства, события и методы сгруппированы по категориям.

Таблица 14. Вкладки панели **Properties**

|  |  |
| --- | --- |
| **Вкладка** | **Описание** |
| All | Содержит список всех свойств в алфавитном порядке |
| Data | Содержит свойства объекта, связанные с источником данных |
| Methods | Содержит список методов объекта |
| Layout | Содержит свойства объекта, связанные с его оформлением |
| Other | Содержит свойства объекта, не вошедшие в предыдущие вкладки |
| Favorites | Позволяет создать список наиболее часто используемых свойств |

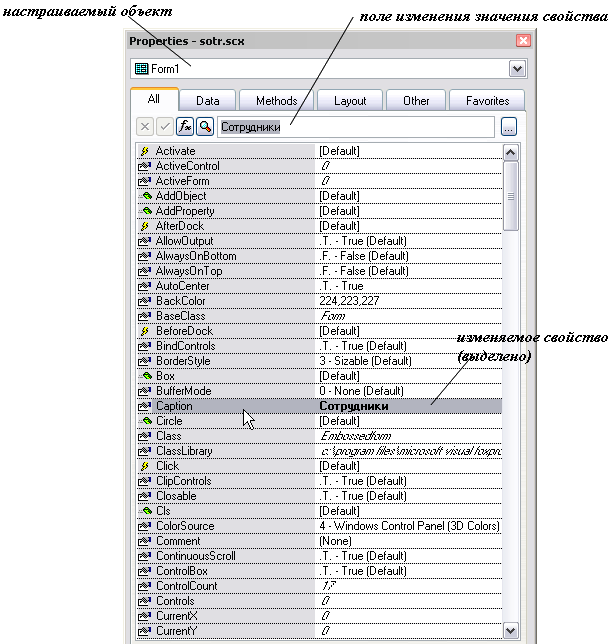


Рис. 28 – Панель Properties

Для изменения значения свойства нужно перейти соответствующее поле и ввести новое значение в поле ввода (вручную, выбором из списка, через вспомогательные окна). Текущее значение свойства отражается справа от его названия.

### ЗАДАНИЕ 6. Редактирование формы с помощью конструктора форм

В результате выполнения задания будет получена форма, представленная на рис. 29.

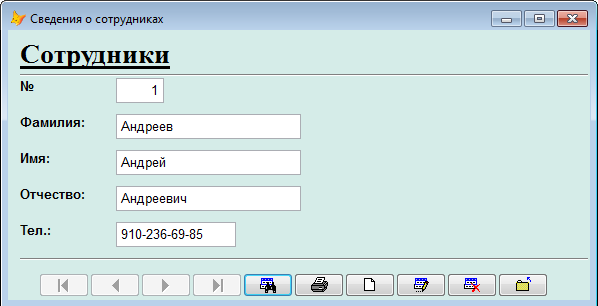


Рис. 29 – Форма Sotr после редактирвания в конструкторе форм

1. Запустите проект **dvd.pjx**.
2. Откройте в режиме конструктора (для редактирования) форму **Sotr**. Для этого выделите ее в менеджере проекта и нажмите кнопку **Modify** (или выполните на имени формы в проекте двойной щелчок левой кнопкой мыши).*Обратите внимание, что теперь все элементы формы доступны для выделения мышью и последующего редактирования (рис. 30).*

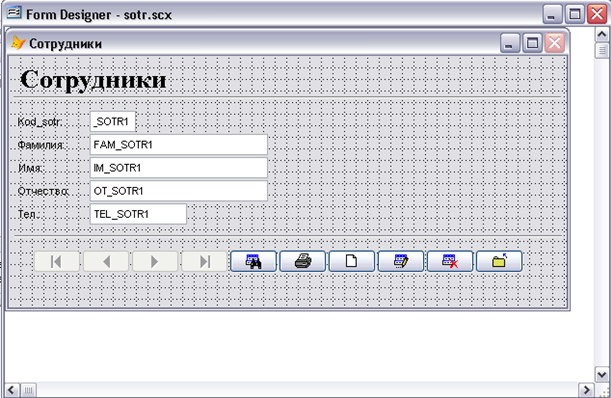


Рис.30 – Форма **Sotr**, открытая в конструкторе форм

1. Если на экране не отображается панель **Properties**, то вызовите на форме контекстное меню и выберите команду **Properties**.
2. Выделите на форме надпись «Сотрудники», выполнив на ней щелчок левой кнопкой мыши. Обратите внимание, что в верхней строке панели свойств отображается объект **Label1**, поэтому все изменения свойств коснутся именно этого объекта.
3. Выполните щелчок левой кнопкой мыши в свободной части формы (на фоне). Обратите внимание, что в верхней строке панели свойств отображается объект **Form1**, т.е. все изменения свойств будут применены именно к форме целиком.
4. В панели свойств выделите свойство **Caption**, выполнив щелчок по соответствующей строке. В поле изменения значения свойства введите новый заголовок формы «Сведения о сотрудниках». Подтвердите изменение нажатием клавиши **Enter**.*Обратите внимание на заголовок формы*.
5. В панели свойство **BackColor**(фон формы) и в поле изменения значения свойства нажмите кнопку . В результате откроется окно выбора цвета. Установите для формы новый цвет. Старайтесь использовать спокойные оттенки цвета, не напрягающие зрительную и психоэмоциональную систему пользователя.
6. Выделите на форме надпись **kod\_sotr**. Убедитесь, что в верхней строке панели свойств отображается объект **LBLKOD\_SOTR1** (это имя надписи, присваиваемое ей программой).
7. Повторяя действия п. 6, измените содержимое надписи (свойство Caption) на**№**.
8. Не снимая выделение с той же надписи, измените ее подчеркивание. Для этого выделите свойство **FontUnderline** и в поле изменения значения свойства выберите из раскрывающегося списка значение **.T. (true)** или выполните двойной щелчок на свойстве.
9. Не снимая выделение с той же надписи, измените ее шрифт (Arial), начертание (полужирное) и размер (10). Для этого на панели Properties выделите свойство  
   **FontCharSet** (настройки шрифта) и выполните по нему двойной щелчок левой кнопкой мыши (или в поле изменения значения свойства нажмите кнопку ).
10. Таким же образом измените настройки оставшихся четырех надписей (Фамилия, Имя, Отчество, Телефон). Для этого предварительно выделите их все (протяжкой мышью или по одному с зажатой клавишей Shift) и повторите действие п. 11.
11. Для всех надписей измените их высоту, установив значение свойства **Height** равным 25. *Скорее всего, после изменения шрифта не все части надписи были видны.*
12. Повторяя действия п. 12, измените настройки шрифта для полей формы **kod\_sotr1**, **fam\_sotr1**, **im\_sotr1**, **ot\_sotr1**, **tel\_sotr1**.
13. Выделите текстовое поле **fam\_sotr1.**На панели Properties выделите свойство  
    **ControlSource**. *Обратите внимание на его значение: sort.fam\_sotr – это поле таблицы базы данных, содержимое которого отображается в поле формы*.
14. Для всех текстовых полей измените их высоту на 25, действуя аналогично п. 13.
15. Пользуясь перетаскиванием мыши, клавишами управления курсором, клавишами управления курсором с зажатой клавишей Ctrl, измените расположение надписей и полей на форме.
16. Закройте форму, сохранив изменения, и проверьте ее работу, запустив нажатием кнопки Run.
17. Повторяя действия п.2-п.18, измените форму Kategor, чтобы она приняла вид, представленный на рис. 31.

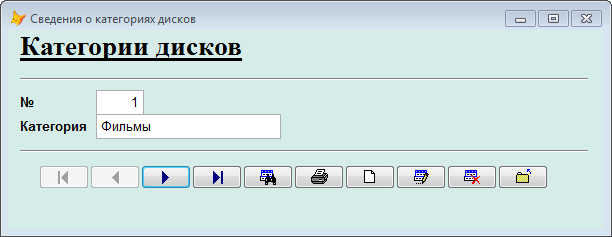


Рис. 31 – Форма Kategor

1. Проверьте работу формы.
2. Повторяя действия п.2-п.18, измените форму Klient, чтобы она приняла вид, представленный на рис. 32.

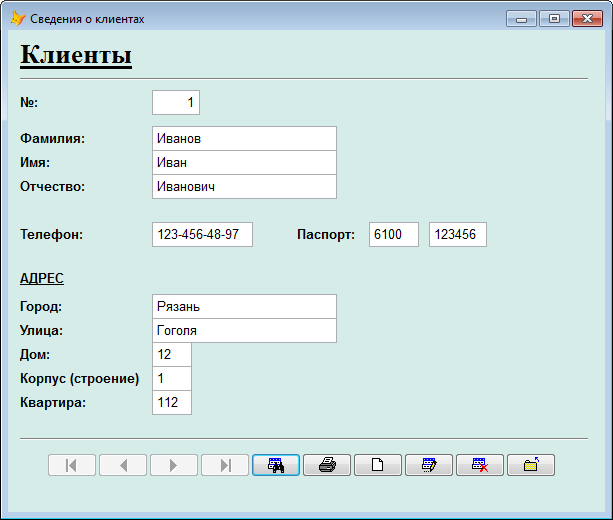


Рис. 32 – Форма Klient

Обратите внимание, что на форме удалена надпись **nom** и добавлена надпись **АДРЕС**. Для добавления надписи выполните следующее:

1. убедитесь, что отображена панель инструментов **Form Controls** (рис. 33), иначе выполните команду меню **View – Form Controls Toolbar**;
2. одинарным щелчком левой кнопки мыши выберите на панели инструмент Labelи выполните щелчок на форме (в том месте, где собираетесь разместить надпись); обратите внимание, что при наведении курсора мыши на объект в панели инструментов появляется его название;
3. выполните настройки надписи аналогично действиям п.4-п.6.



Рис. 33 – Панель инструментов **Form Controls**

1. Проверьте работу формы.
2. Завершите работу с проектом.

### ЗАДАНИЕ 7. Создание формы с помощью конструктора форм

В результате выполнения задания будет получена форма, представленная на рис. 34.

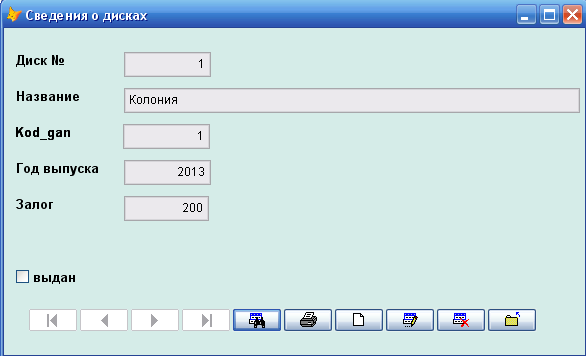


Рис. 34 – Форма **Disk**, созданная с помощью конструктора форм

1. Запустите проект **dvd.pjx**.
2. Запустите конструктор форм, чтобы создать форму на основе таблицы Disk.dbf. Для этого перейдите на вкладку **Documents (Docs)** менеджера проекта, выделите объект **Forms** и нажмите кнопку **New**. В появляющемся окне запроса способа создания выберите **NewForm**. В результате будет запущен конструктор форм (рис. 35).

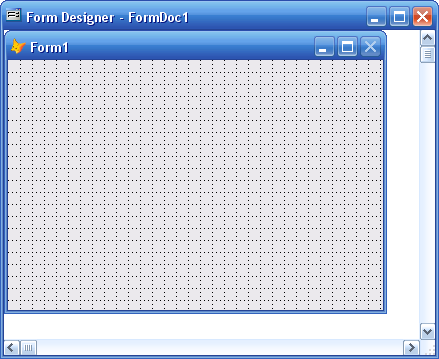


Рис. 35 – Окно конструктора форм

1. Убедитесь, что отображена панель инструментов **Form Controls** (рис. 33), иначе выполните команду меню **View – Form Controls Toolbar**.
2. Измените заголовок формы на «Сведения о дисках» (Задание 6, п.5 – п.6).
3. Добавьте источник данных для формы:
   1. вызовите контекстное меню на свободной области формы и выберите команду **DataEnvironment**;
   2. в появляющемся окне **AddTableorView** (рис. 36) убедитесь, что в верхней строке в списке отображается имя вашей базы данных, выделите таблицу **Disk** и нажмите кнопку **Add** (или выполните двойной щелчок левой кнопкой мыши на имени добавляемой таблицы); закройте окно добавления, нажав кнопку **Close**.*Обратите внимание, что окно* ***DataEnvironment****все еще открыто*.

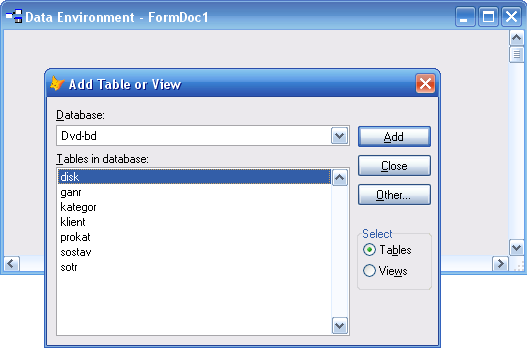


Рис. 36 – Окно добавления источника данных для формы

1. В окне источника данных выделите поле **kod\_dis** и перетащите его на область формы.
2. Аналогичным образом перетащите на форму поля **nazv**, **kod\_gan**, **god, v\_prok**.
3. Закройте окно **Data Environment**.
4. Добавьте поле **zalog** с помощью панели **FormControls** (рис. 33):
   1. выделите на панели инструмент **TextBox** и выполните щелчок в свободной области формы (появится поле с именем **Text1**);
   2. свяжите созданное поле с полем **zalog** таблицы базы данных **disk**: выделите поле **Text1** и на панели **Properties** в свойстве **ControlSource** выберите значение **disk.zalog**.
5. Добавьте на форму панель кнопок для управления записями. Для этого в менеджере проекта перейдите на вкладку **Classes**, раскройте узел **wizbtns**, выберите объект **picbtns** (панель кнопок с пиктограммами) или **txtbtns** (панель кнопок с надписями) и перетащите его на форму.
6. В результате форма примет вид, представленный на рис. 37:

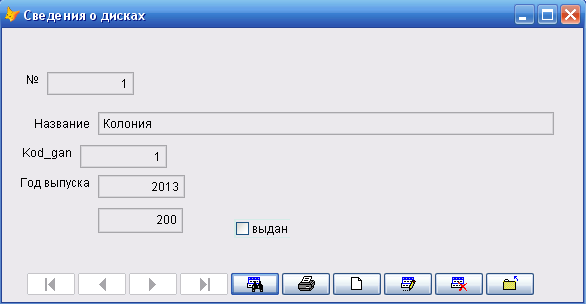


Рис. 37 – Внешний вид формы

1. Отредактируйте форму так, чтобы она приняла вид, представленный на рис. 34.
2. Запустите созданную форму.
3. Обратите внимание, что поле **v\_prok** представлено флажком «Выдан» (элемент CheckBox).
4. Отредактируйте первую запись, пометив первый диск, как выданный с помощью флажка «выдан».
5. Завершите работу с проектом.

## 4.3. Размещение на форме элемента ComboBox

**Combo Box** (раскрывающийся список) –это список, который в обычном состоянии свернут, но имеет кнопку для раскрытия списка (рис. 38).

Несомненным плюсом ComboBox является его экономичность, т. к. он занимает мало места на форме. При нажатии на кнопку список раскрывается, предоставляя возможность выбрать из списка одно из значений.

|  |  |
| --- | --- |
|  |  |

Рис. 38 – Объект **ComboBox** на базе таблицы **Ganr**

Данный объект позволяет не просто выбирать одно из значений списка, но и выбирать и отображать значение одного столбца списка, а сохранять в качестве выбранного значение другого столбца этого же списка. Например, как показано на рис. 38, выбирается и отображается значение столбца **ganr**, а в качестве результата выбора сохраняется значение столбца **kod\_gan**. Выбранное значение записывается в свойство **Value**, а отображаемое – в свойство **DisplayValue**.

**ComboBox** не предназначен для ввода и редактирования элементов списка. Хотя можно и добавлять, и модифицировать, но это требует дополнительных усилий от разработчика, которые связаны с разработкой программной процедуры.

Чаще всего раскрывающийся список содержит данные из некоторой таблицы БД и служит для заполнения полей внешних ключей в связанных с ними таблицах. Хотя возможно указать в качестве элементов раскрывающегося списка некоторый набор константных значений.

**ComboBox** имеет соответствующий построитель (**Builder**), который позволяет выполнить его настройки. Использование построителя будет изучено при выполнении Задания 7.

В некоторых ситуациях невозможно применить **Builder** для **ComboBox** (например, когда **ComboBox** размещается на форме в ячейке таблицы). Тогда для настройки **ComboBox** следует использовать панель **Properties**. Вообще говоря, ComboBox обладает широким набором свойств, но будут рассмотрены только те из них, которые непосредственно влияют на работу списка и используются чаще всего при написании процедур.

**Некоторые свойства ComboBox:**

**RowSourceType** тип источника данных (обычно это таблица базы данных);

**RowSource** источник данных для списка (поля, которые появятся при раскрытии списка; на первом месте следует указать поле, которое будет отображено в результате выбора и сохранится в свойстве **DisplayValue**);

**BoundColumn** номер столбца списка, значение из которого следует сохранять (обычно это первичный ключ таблицы, на базе которой строится список; это значение будет записано в свойство **Value**);

**ControlSource** поле, в котором будет сохранено значение (обычно это внешний ключ таблицы, связанной с таблицей, выступающей в качестве источника данных для списка);

**BoundTo** показывает, что сохранять (рис. 39): выбранное значение (**.T.**) или его порядковый номер (**.F.**); например, при установленном значении свойства **.T.** сохраняется число 4, а при установленном значении свойства **.F.** сохраняется число 3 (что, вообще говоря, некорректно и нарушит достоверность сведений в базе данных);

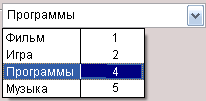


Рис. 39 – Раскрывающийся список

**Value** выбранное значение;

**DisplayValue** отображаемое значение;

**ColumnWidths** ширина столбцов (например, установка ширины второго столбца равной нулю позволит скрыть его от пользователя, но выполнять корректный выбор).

По сути, для настройки корректной работы **ComboBox** достаточно изменить первые 5 свойств из списка.

### Задание8. Размещение в форме ComboBox и его настройка с помощью Builder

1. Запустите проект **dvd.pjx**.
2. Запустите форму **Ganr.scx**. Просмотрите записи с помощью кнопок перехода по записям. *Обратите внимание, каким образом для каждого жанра определяется соответствующая категория – при помощи кода категории. Естественно, данный способ не удобен и не нагляден, поскольку, чтобы отнести жанр к какой-либо категории, необходимо воспользоваться таблицей Kategor и посмотреть, какой код соответствует той или иной категории, а уж потом указать его для каждого жанра. Поэтому при добавлении новых жанров целесообразно выбирать соответствующую категорию из списка (поскольку перечень категорий уже хранится в базе данных).*Закройте форму.
3. Откройте ту же форму для редактирования в режиме конструктора (ЗАДАНИЕ 6, п.2).
4. Прежде всего, добавьте к источникам данных для формы таблицу базы данных, на основе которой будет создаваться список – таблицу **Kategor.dbf**:
   1. вызовите контекстное меню на свободной области формы и выберите команду **DataEnvironment**;
   2. в появляющееся окно источника данных добавьте таблицу **Kategor.dbf**, для чего вызовите контекстное меню на свободной области окна **DataEnvironment** и выберите команду **Add**; в результате на экране отобразится окно **AddTableorView** (рис. 36);
   3. добавьте таблицу в источник данных (ЗАДАНИЕ 7, п.5-б).
5. В окне **DataEnvironment** выделите связь между таблицами и удалите ее нажав клавишу **Delete**;измените связь, протянув ее от таблицы *Ganr* к таблице **Kategor** (рис. 40).

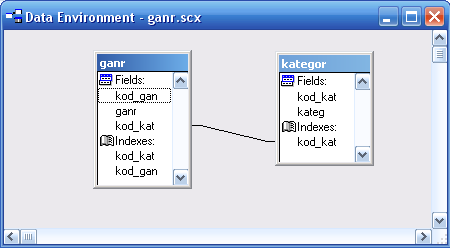


Рис.40 – Добавление таблицы-основы для раскрывающегося списка

1. Добавьте на форму раскрывающийся список, в котором будут отображаться имеющиеся категории дисков:
   1. на панели инструментов **FormControls** выберите объект **ComboBox** (кнопка );
   2. выполните щелчок в свободной области формы (или размесите объект, создавая на свободной области формы рамку, ограничивающую будущий список).
2. Выполните настройки списка с помощью **Builder**:
   1. вызовите контекстное меню на **Combo1** и выберите команду **Builder**;
   2. выберите столбцы таблицы базы данных, которые станут содержимым списка (рис. 41):

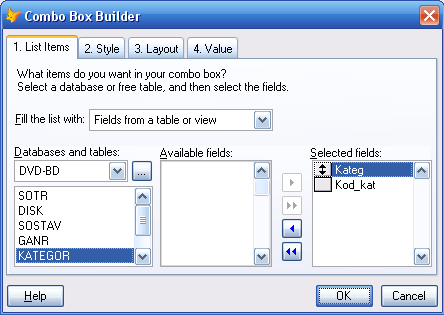


Рис.41 – Выбор содержимого списка

*Обратите внимание: поле Kateg указано на первом месте, т.к. именно оно будет отображаться при выборе категории из спика. Поле Kod\_kat будет использоваться для сохранения результата выбора в таблице Ganr, поэтому добавьте и его.*

* 1. настройте сохранение результата выбора (рис. 41):

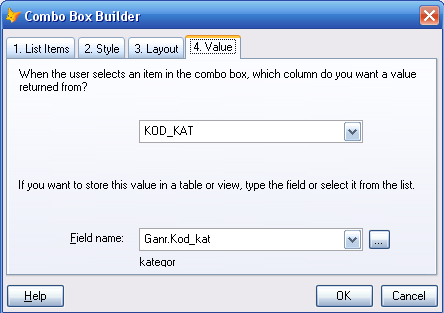


Рис. 41 – Настройка сохранения результатов выбора

*Обратите внимание: в первом поле выбирается ЧТО сохранять (первичный ключ таблицы Kategor), во втором – КУДА сохранять (соответствующий внешний ключ таблицы Ganr).*

* 1. завершите работу с построителем.

1. Завершите редактирование формы и запустите ее. Перемещаясь по записям, убедитесь, что для каждого жанра корректно отображается соответствующая категория.
2. С помощью отредактированной формы добавьте новый жанр «Классика» (будет введен в поле Жанр) к категории «Музыка» (будет выбрана из созданного списка), при этом *поле kod\_kat заполнять не нужно* (как и kod\_gan). Сохраните изменения. Обратите внимание, что в поле *kod\_kat*автоматически появляется номер выбранной категории «Музыка».
3. Откройте форму в режиме конструктора, выделите созданный список Combo1 и обратите внимание на значения свойств объекта в панели Properties на вкладке **Data**:
   1. **BoundColumn** – 2
   2. **ControlSource** – Ganr.Kod\_kat
   3. **RowSource** – kategor.kateg,kod\_kat
   4. **RowSoursType** – 6-Fieldes

Именно эти настройки были выполнены на вкладках **Builder**.

1. Установите свойство **BoundTo**=**.T.**.
2. Отредактируйте форму так, чтобы она приняла вид, представленный на рис. 42:

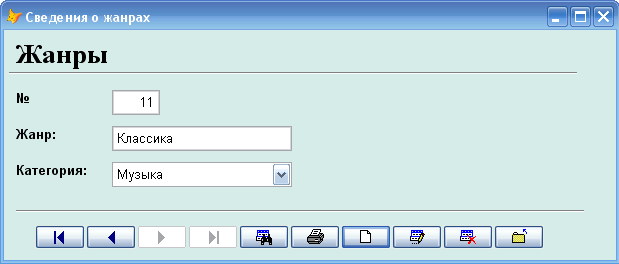


Рис. 42 – Итоговый вид формы Ganr

1. Сохраните форму, проверьте ее работоспособность.
2. Завершите работу с проектом.

### Задание9. Размещение в форме ComboBox и его настройка с помощью панели Properties

1. Запустите проект **dvd.pjx**.
2. Откройте для редактирования в конструкторе форму Disk.
3. Создайте на ней раскрывающийся список на основе таблицы Ganr (чтобы при внесении в базу данных новых дисков не вводить вручную код соответствующего жанра, а выбирать сам жанр из списка жанров, уже хранящихся в базе данных):
   1. добавьте в источники данных таблицу Ganr (ЗАДАНИЕ 8, п.4);
   2. измените связь между таблицами Disk и Ganr (ЗАДАНИЕ 8, п.5);
   3. с помощью панели FormControls разместите на форме элемент ComboBox (ЗАДАНИЕ 8, п.6);
   4. на панели Proprties для выделенного Combo1 установите следующие свойства:
      1. **RowSoursType** – 6-Fieldes
      2. **RowSource** – ganr.ganr,kod\_gan
      3. **BoundColumn** – 2
      4. **BoundTo**– .T.
      5. **ControlSource** – Disk.kod\_gan
4. Удалите надпись «kod\_gan» и соответствующее ей поле. На их месте разместите надпись «Жанр» и созданный список.
5. Проверьте работоспособность формы и добавьте с ее помощью еще один диск:
   1. название – Дивергент;
   2. жанр – Фантастика;
   3. год выпуска – 2014;
   4. залог – 300.
6. Завершите работу с проектом.

## 4.4. Размещение в форме объектаGrid

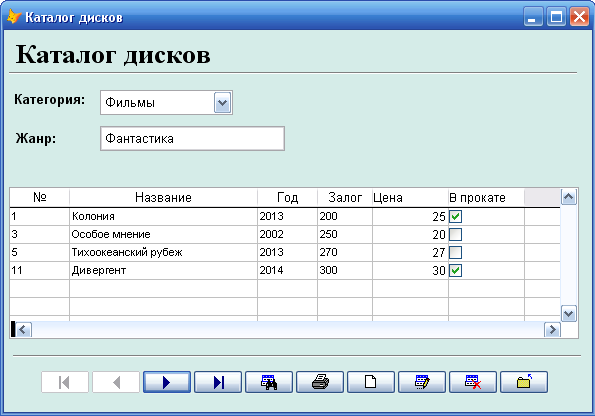


Рис. 43 – Форма с элементом **Grid**

Объект**Grid** (Таблица) служит для просмотра и модификации только табличных данных (в т. ч. временных таблиц – курсоров).

**Grid** – это набор объектов, позволяющий представлять данные в виде таблицы-списка, который можно просматривать в двух направлениях при помощи полос прокрутки.

Благодаря наглядности в **Grid**, намного упрощается проведение расчетов с хранящимися в таблице данными.

Объект**Grid**, несмотря на свою внешнюю простоту, является одним из самых сложных элементов управления формы. Сложен он с точки зрения понимания его структуры и работы.

**Составляющие Grid**

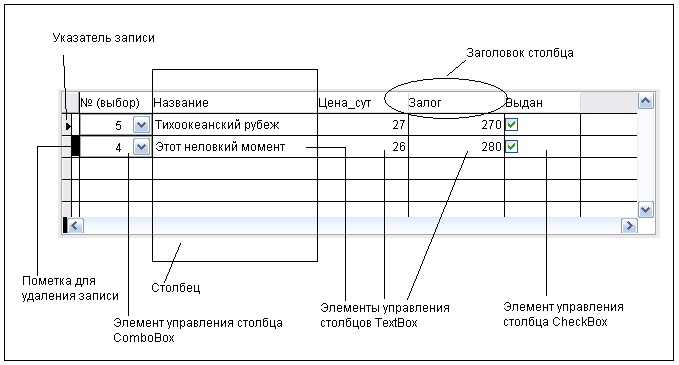


Рис. 44 – Составляющие объекта **Grid**

1. Строки и столбцы (**Column**).
2. Полосы прокрутки.
3. Заголовок (**Header**) каждого столбца.
4. Элемент управления каждого столбца. По умолчанию – это текстовое поле (**TextBox**), но данный элемент может быть и другим, например, **ComboBox**, **CheckBox**.
5. Указатель записи.
6. Пометка для удаленных записей.

На практике **Grid** чаще всего используется при создании форм, в которых отображается содержимоелибо одной таблицы базы данных (тогда вид представления данных похож на результат работы команды **Browse**), либо двух таблиц базы данных, связанных отношением «один-ко-многим». В большинстве случаев именно второй вариант применения наиболее распространен, т.к. представляет более наглядным и удобным образом отобразить информацию, содержащуюся в соответствующих таблицах.

**Размещение объекта Grid на форме** может быть выполнено двумя способами:

1. Добавление с помощью панели **FormControls** в конструкторе форм. Данный способ предпочтительней использовать в том случае, когда **Grid** будет использоваться на форме, отображающей содержимое только одной таблицы базы данных. Этот способ не очень удобен (хотя и может быть использован) в тех случаях, когда **Grid** будет использоваться для отображения содержимого взаимосвязанных таблиц, поскольку его настройка в режиме конструктора будет достаточно трудоемкой.
2. Создание формы на основе двух взаимосвязанных таблиц с помощью мастера форм. При таком использовании мастера форм **Grid** появится на форме автоматически, при этом в процессе работы мастера можно будет выбрать набор полей, образующих столбцы **Grid**. Между основной частью формы и объектом **Grid** будет установлена связь в соответствии с базой данных. Ввиду таких преимуществ, именно такой способ размещения **Grid** на форме является более предпочтительным. Каковы особенности получающейся формы? На рис. 43 приведен пример формы с **Grid**. Форма создана на основе таблиц **Ganr** и **Disk**, связанных отношением «один-ко-многим»: **Ganr****Disk**. Содержимое многосвязной сущности (**Disk**) отображается в **Grid**, а односвязной (**Ganr**) – в основной части формы.

В столбцах объекта **Grid** могут содержаться данные из одной или нескольких таблиц базы данных.

**Grid** имеет встроенный построитель, который вызывается командой **Builder** из контекстного меню **Grid**. Построитель позволяет:

1. Изменить перечень полей таблицы базы данных, которые являются столбцами **Grid**.
2. Изменить стиль оформления **Grid**.
3. Изменить заголовки столбцов и элементы управления столбцов.
4. Изменить связь между таблицами, если **Grid** появилась на форме, созданной на основе двух связанных таблиц.

***Построитель следует использовать только в том случае, если в столбцах Grid отображаются данные лишь одной таблицы базы данных****.*

**Свойства Grid и ее компонент**

**Grid** является сложным объектом формы и имеет определенную структуру. Причем выделяют отдельно свойства **Grid** как одного объекта в целом, а отдельно свойства каждого ее компонента. Перечень и тех, и других свойств достаточно обширен, поэтому остановимся на свойствах, используемых при выполнении заданий пособия.

**Некоторые свойства Grid**

**BackColor** цвет заливки ячеек таблицы.

**ColumnCount** изменение количества столбцов. Позволяет удалять последние столбца таблицы. Например, если **Grid** имела 6 столбцов, то при изменении свойства на 5 будет удален последний столбец **Column6**.

**HeaderHeight** высота строки заголовков (может быть изменена путем перетаскивания разделителя, аналогично изменению высоты строк в Excel).

**Height** высота таблицы (может быть изменена путем перетаскивания границ **Grid**, аналогично изменению размера окна в Windows).

**RecordSource** источник данных (таблица).

**RowHeight** высота строк (может быть изменена путем перетаскивания разделителя, аналогично изменению высоты строк в Excel).

**ScroolBars** выбор полос прокрутки (нет, вертикальная, горизонтальная, обе).

**RecordSourceType** тип источника данных для **Grid** (обычно, 1 – Alias).

**RecordSource** источник данных (обычно таблица со стороны многосвязной сущности).

Существует большое количество свойств оформительского характера (шрифты, цвета), изучение которых не предоставляется затруднительным.

Чтобы изменять свойства отдельных частей **Grid**, следует выделить требуемый компонент. Выделение доступно после перехода в режим редактирования **Grid** (команда **Edit** контекстного меню **Grid**). Также можно выбрать компонент для настройки в верхней строке панели **Properties**.

**Некоторые свойства столбцов Grid (Column)**

**ControlSource** Источник данных для столбца (поле таблицы или формула, зависящая от полей таблиц)

**CurrentControl** Элемент управления столбца (**Text1**, **Combo1**, **Check1**)

**Sparse** Элемент управления отображается в текущей строке **Grid** (.**T**.) или во всех столбцах (.**F**.).

**Некоторые свойства заголовков столбцов Grid (Header)**

**Caption** Содержимое заголовка столбца

Основные компоненты **Grid** отображаются в верхнем списке панели **Properties** (рис. 45). При изменении свойств **Grid** или ее частей следует контролировать содержимое этого поля.

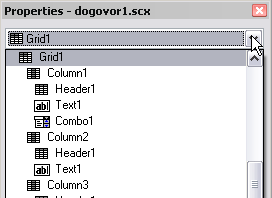


Рис. 45 – Список компонент **Grid** на панели **Properties**

1. Непосредственно **Grid**.
2. Входящие в ее состав столбцы: **Column1**, **Column2** и т.д. (визуально в списке столбцы расположены правее уровня соответствующей **Grid**)
3. Внутри каждого столбца:
   1. **Header1** – заголовок соответствующего столбца.
   2. **Text1**, **Combo1** – элементы управления, расположенные в ячейках этого столбца (визуально эти элементы расположены правее уровня соответствующего столбца).

### ЗАДАНИЕ 10. Создание формы с объектом Grid

1. Запустите проект **dvd.pjx**.
2. Создайте с помощью мастера форму на основе таблиц GanrDisk. Для этого из проекта запустите мастер форм, выберите вариант создания **One-to-ManyFormWizard** и следуйте шагам мастера:
   1. выбор родительской таблицы (рис. 46); ею становится таблица, данные которой будут располагаться в «шапке» формы, т.е. таблица **Ganr**

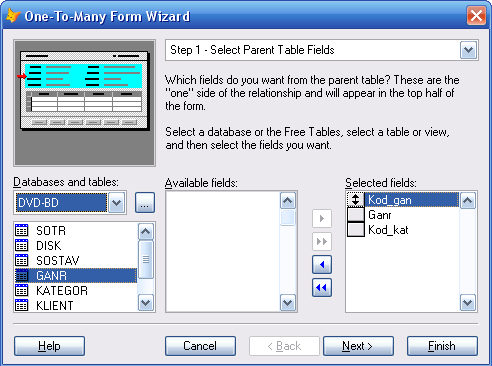


Рис. 46 – Выбор родительской таблицы

* 1. выбор подчиненной таблицы (рис. 47); ею становится таблица, которая на форме станет табличной частью Grid, т.е. таблица Disk (*обратите внимание, что поля Cena, Zalog,v\_prok не добавлены*)

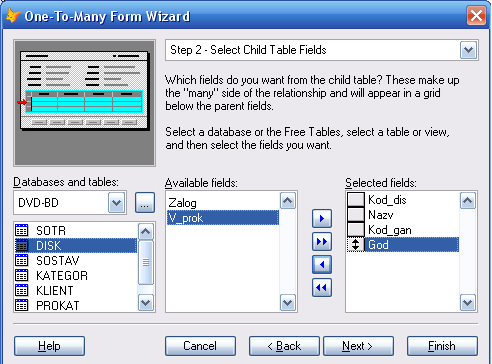


Рис. 47 – Выбор подчиненной таблицы

* 1. установка связи между таблицами (рис. 48);при правильно созданной базе данных эта связь и соответствующие ключи выставляются автоматически

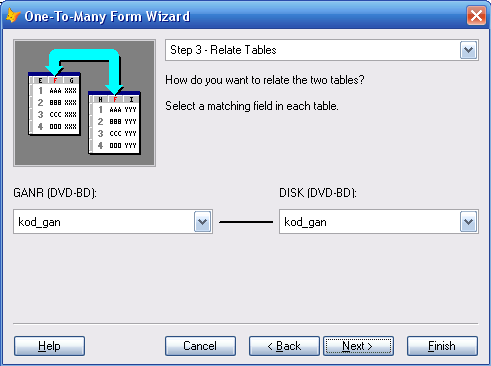


Рис. 48 – Установка связи между таблицами

* 1. выбор стиля формы (рис. 22); установки шага аналогичны мастеру форм на основе одной таблицы;
  2. выбор поля для сортировки (рис. 23); установки шага аналогичны мастеру форм на основе одной таблицы;
  3. завершение создания формы (рис. 24); установки шага аналогичны мастеру форм на основе одной таблицы.

1. Сохраните форму под именем **Ganr-Disk.scx**.
2. Запустите созданную форму (рис. 49). Данные подчиненной таблицы (Disk) отображаются в Grid, а данные главной (родительской) таблицы – в основной части формы.

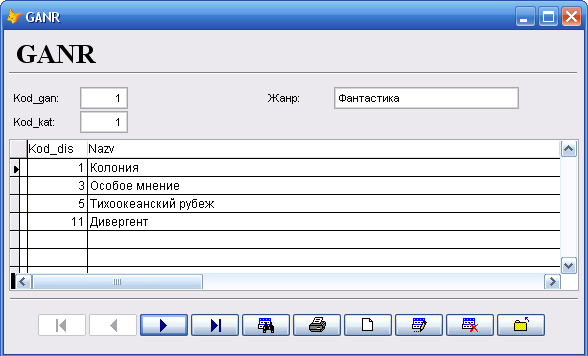


Рис. 49 –Форма **Ganr-Disk.scx**

1. Используя горизонтальную линейку прокрутки, убедитесь, что табличная часть (**Grid**) содержит все указанные во втором шаге мастера столбцы.
2. Откройте созданную форму в режиме конструктора.
3. Увеличьте ширину **Grid**. Для этого выделите ее, в результате чего **Grid** будет обрамлена рамкой с шестью прямоугольными маркерами размера. Используя их, увеличьте ширину **Grid** при помощи перетаскивания их мышью.
4. Внесите в **Grid** несколько изменений с помощью построителя. Для этого вызовите на **Grid** контекстное меню и выберите команду **Builder**.
   1. на первой вкладке добавьте в **Grid** столбец**zalog** переместите столбец **kod\_gan** на последнее место, используя маркер  (рис. 50);

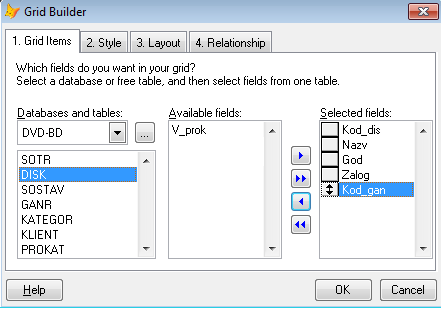


Рис. 50 – Окно построителя **Grid**, изменение набора столбцов

* 1. на второй вкладке измените стиль оформления **Grid** (рис. 51);

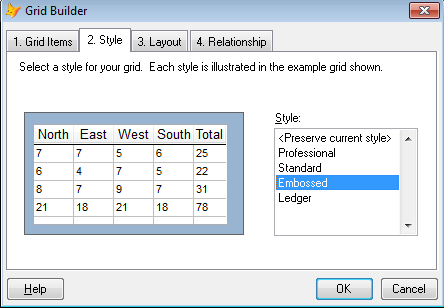


Рис. 51 – Окно построителя Grid, изменение стиля оформления

* 1. на третьей вкладке измените заголовок столбца **kod\_dis** на **№**, выполнив щелчок по столбцу ниже заголовка и введя новое значение в поле **caption** (рис. 52); здесь же уменьшите ширину столбца **nazv**, подведя курсор мыши к правой границе столбца в строке с заголовком, пока он не примет вид двунаправленной стрелки, и перетащив курсор;

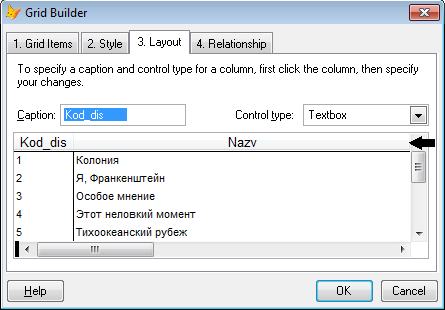


Рис. 52 – Окно построителя **Grid**, изменение подписей  
и управляющихэлементов столбцов

* 1. аналогичным образом измените заголовок столбца **nazv** на **Название**;
  2. обратите внимание на содержимое четвертой вкладки – здесь указаны поля, по которым реализуется связь между родительской и подчиненной таблицами (рис. 53).



Рис. 53 – Окно построителя **Grid**, связь между таблицами

1. Завершите работу с построителем и отредактируйте размеры формы и **Grid** так, чтобы отображались все ее столбцы.
2. Запустите форму, она примет вид, представленный на рис. 54. Обратите внимание, что по сути данная форма имеет то же назначение, что и форма **disk.scx**, но используется другой вариант оформления.

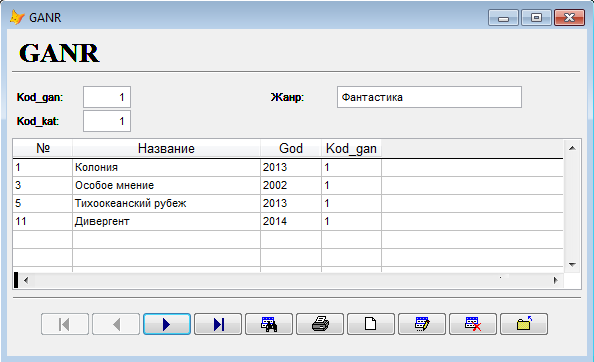


Рис. 54 –Отредактированная форма

1. Завершите работу с проектом.

### ЗАДАНИЕ 11. Изменение Gridс помощью панели Properties

Как уже было отмечено, не всегда настройка Grid может быть выполнена с помощью построителя, поэтому следует использовать панель Properties (в некоторых случаях ее использование оказывается более удобным).

1. Запустите проект **dvd.pjx**.
2. Откройте форму**ganr-disk.scx** в режиме конструктора.
3. Перейдите на панель Properties и в списке выбора объектов найдите объект Grid1. Обратите внимание, что в списке под Grid отображаются ее составляющие, как представлено на рис. 45.
4. В контекстном меню **Grid** выберите команду **Edit** – режим редактирования таблицы. Обратите внимание на то, что вокруг Grid появилась голубая рамка выделения.
5. Выполните щелчок внутри столбца **God**. Обратите внимание, что в панели **Properties** в строке выбора объекта стал отображаться **Column3** (это и есть столбец God). *Теперь все изменения свойств в панели Properties будут касаться именно этого столбца.****Обращайте внимание на содержимое верхней строки панели Properties, чтобы менять свойства нужного объекта****.*
6. Обратите внимание на значение свойства **ControlSource** – **disk.god** – это источник данных для столбца.
7. Обратите внимание, что в перечне свойств есть уже ранее использованные свойства для настройки шрифта: **FontCharSet**, **FontSize**, **FontName** и т.д.
8. Установите значение свойства **Width** равным 40 – изменилась ширина столбца God.
9. Оставаясь в режиме редактирования таблицы (**Edit**), выполните щелчок на заголовке столбца **God**. Обратите внимание, что в панели **Properties** в строке выбора стал отображаться объект **Header1** (при необходимости может потребоваться выполнить щелчок повторно). Раскройте список объектов панели **Properties** и обратите внимание, что выделенный объект **Header1** относится к группе столбца **Column3** (рис. 55).

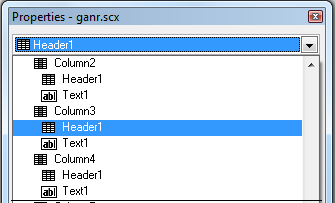


Рис. 55 – Выбор заголовка столбца **Grid**

1. Введите в свойство **Caption** значение **Год**. Обратите внимание на то, что изменился заголовок столбца.
2. Аналогичным образом измените заголовок столбца**Zalog** – на **Залог**.
3. Выберите столбец **kod\_gan** (аналогично п.5) и установите свойство **Visible** равным **.F. – False** (значение «ложно»). Обратите внимание, что столбец перестал отображаться.
4. Немного увеличьте высоту строк, подведя указатель мыши к левой к границе таблицы изнутри к нижней границе первой строки, пока указатель не примет вид двунаправленной стрелки, и перетащите ее. Таким же способом можно менять и ширину столбцов, используя границу между заголовками. Перетаскиванием заголовков можно менять порядок расположения столбцов.
5. Щелкните в свободной области формы, чтобы выйти из режима редактирования таблицы (**Edit**). Выделите таблицу. Обратите внимание на значение свойства **ColumnCount** – 5. Это количество столбцов таблицы. Изменяя это количество, можно добавлять или удалять столбцы таблицы.
6. Добавьте в таблицу еще один столбец, в котором пока будет отображаться величина залога диска (**disk.zalog**), в дальнейшем она будет изменена на цену за сутки проката. Для этого следует увеличить количество столбцов на 1, т.е. установите значение свойства **ColumnCount**равным 6. Обратите внимание, что в таблице появился новый столбец с заголовком **Header1**. Такие же изменения произошли и в списке объектов в панели **Properties**.
7. Свяжите добавленный столбец с полем **disk.zalog**. Для этого выберите добавленный столбец (аналогично п.4 – п.5) и в панели **Properties** в свойстве **ControlSource** выберите **disk.zalog**.
8. Измените заголовок добавленного столбца на**Цена** (аналогично п.11).
9. Запустите отредактированную форму и убедитесь в правильности выполнения задания.
10. Завершите работу с проектом.

### ЗАДАНИЕ 12. Изменение управляющего элемента столбцаGrid

По умолчанию каждая ячейка **Grid** является текстовым полем, но достаточно часто возникает необходимость отображать в столбцах флажки (**CheckBox**), раскрывающиеся списки (**ComboBox**) и другие элементы управления (рис. 44).

1. Запустите проект **dvd.pjx**.
2. Запустите форму **disk.scx**, обратите внимание на **первую** запись – диск отмечен, как выданный, с помощью флажка.
3. Откройте в конструкторе форму **Ganr-disk.scx**.
4. Повторяя действия п.15 – п.16 ЗАДАНИЯ 11, добавьте в Grid еще один столбец для отображения поля **disk.v\_prok**.
5. Запустите отредактированную форму, обратите на значение поля **disk.v\_prok** для **первого** диска. Здесь отображается значение **T** – истина, а не флажок.
6. Снова откройте форму в конструкторе и измените тип столбца v\_prok так, чтобы в нем отображался флажок, а не логическое значение:
   1. перейдите в режим редактирования Grid (Задание 11, п.4);
   2. на панели **FormControls** выберите объект **CheckBox** и выполните щелчок в столбце **v\_prok** в любом месте под заголовком – объект добавлен в Grid (в панели **Proprties** появился объект **Check1**), но *визуально в таблице никаких изменений не видно*;
   3. измените тип управляющего элемента для столбца **v\_prok**: выберите столбец, выполнив щелчок внутри него под заголовком (убедитесь, что в панели **Properties** в списке выбора объектов отображается **Column7**) и установите значение свойства **CurrentControl** равным **Check1**; *обратите внимание, что в столбце таблицы появился флажок*;
   4. Чтобы вставленный объект (**Check1**) отображался не а текущей, а во всех строках Grid, изменитe свойство **Sparse**, установив значение **.F. – False**.
7. Запустите созданную форму, чтобы убедиться в правильности добавления флажка. *Обратите внимание, что рядом со значком флажка расположена надпись* ***Check1***. Чтобы ее удалить, откройте форму в конструкторе, в панели **Properties** отыщите добавленный флажок (рис. 56) и измените для него свойство **Caption**, сделав его пустым.

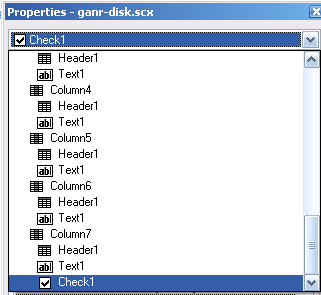


Рис. 56 – Флажок в качестве управляющего элемента столбца

1. Проверьте работоспособность формы и с ее помощью отметьте диск с №11 как выданный.
2. Отредактируйте форму так, чтобы она приняла вид, представленный на рис. 57. *Обратите внимание, что на форме появляется раскрывающийся список с перечнем категорий, создание которого было описано в ЗАДАНИИ 8.*

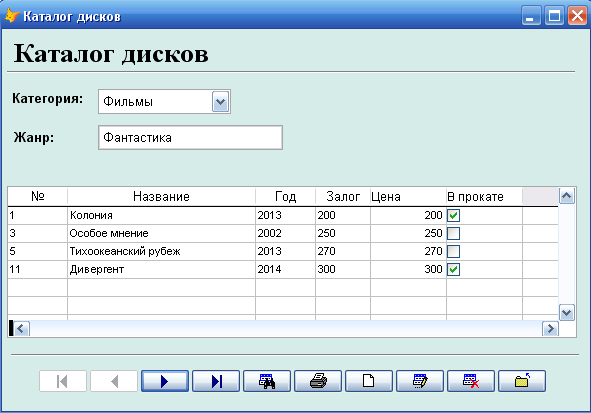


Рис. 57 – Итоговый вариант формы Ganr-disk.scx

1. Созданная форма позволяет добавлять жанры выбранной категории и диски этого жанра.
2. Добавьте с помощью данной формы жанр **Боевик** в категорию **Фильмы**. Для этого запустите форму и нажмите кнопку **Add/Saverecord**. В результате появится диалоговый запрос параметров добавления данных, представленный на рис. 58.



Рис. 58 – Запрос на добавление записи

Окно позволяет определить, в какую таблицу (из тех двух, на основе которых построена форма) будут добавлены записи:

* 1. **Addrecordtoparentonly** – добавить одну запись а главную таблицу («шапка» формы, т.е. таблица Ganr);
  2. **Addrecordtochild (grid) only** – добавить одну запись в подчиненную таблицу, т.е. в Grid (Grid, табличная часть, т.е. таблица Disk);
  3. **Addrecordtoboth** – добавить одну запись в обе таблицы (добавить одну запись в «шапку» формы (т.е. в таблицу Ganr) и одну соответствующую ей запись в Grid (т.е. в таблицу Disk)).

1. Т.к. будет выполняться добавление данных в основную часть формы (Ganr), то в окне запроса выберите первый переключатель. Выберите нужную категорию и добавьте требуемый жанр.
2. Теперь добавим для жанра Боевик запись о фильме. Для этого повторите действия п.11 и в окне запроса выберите второй переключатель, т.к. будет выполнено добавление только в таблицу Disk. Добавьте данные о фильме:



1. Действуя аналогично, добавьте еще одну запись о фильме того же жанра:



1. Завершите работу с проектом.

### ЗАДАНИЕ 13. Создание вычисляемого столбца в Grid

Создание вычисляемого столбца в Grid выполняется за счет добавления нового столбца изменения его свойства **ControlSource**, в котором и указывается выражения для вычисления.

1. Запустите проект **dvd.pjx**.
2. Откройте в режиме конструктора форму **Ganr-disk.scx**.
3. Выделите столбец «Цена» (аналогично п.4 – п.5 ЗАДАНИЯ 11 или выберите **Column6** в панели **Properties**) и измените значение свойства **ControlSource** на **disk.zalog\*0.1**.
4. Запустите форму и убедитесь в правильности вычислений.
5. Завершите работу с проектом.

## 4.5. Размещение на форме вычисляемых полей

Возможность создавать на форме вычисляемые поля – несомненное достоинство VisualFoxPro. Именно в вычисляемых полях формы появятся поля, удаленные на этапе нормализации базы данных. В вычисляемых полях могут появиться результаты как простых расчетов, так и достаточно сложных процедур. Вообще говоря, с каждой формой может быть связано значительное количество вычислений разной степени сложности. Каждое такое вычисления представляет собой процедуру, аналогичную привычным программам, написанным с помощью языков программирования. В принципе VisualFoxPro – это тоже среда программирования, и существует большое количество разнообразных управляющих конструкций, применяемых для создания программ. Освоение этих конструкций не предоставляется сложным для пользователя, имеющего навыки программирования в принципе.

Итак, для вычислений на формах придется разрабатывать программные процедуры. Но как же запускать их на выполнение? Запуск каждой процедуры будет инициирован тем или иным событием, связанным с формой или ее элементом. Событие – это определенное состояние объекта, это своего рода сигнал, по которому выполняются указанные вами действия (процедуры).

Итак, процедуру будут связаны с событиями объектов, они будут запускаться при появлении того или иного события. Но возникает еще один вопрос: событие какого объекта выбрать? Принципиально важно верно определить объект, от которого зависит процедура, и событие, связанное с этим объектом.

Форма и каждый ее объект обладает значительным перечнем событий, и не представляется физически возможным рассмотреть каждое из них. Поэтому будет рассматривать события, используемые при выполнении заданий пособия.

Итак, определены процедуры, определены объекты и их события, которые эти процедуры будут запускать. Как же написать саму процедуру. Во-первых, следует открыть окно написания кода процедуры. Это нужно сделать в конструкторе формы, выполнив двойной щелчок левой кнопкой мыши по форме или объекту. В результате будет открыто окно, представленное на рис. 59.

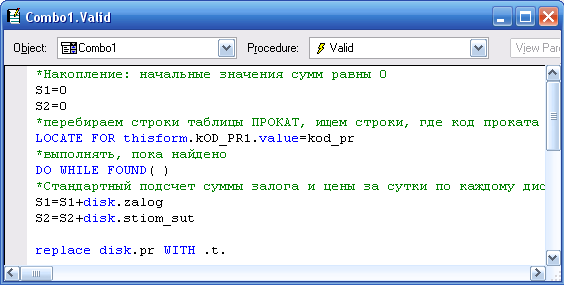


Рис. 59 – Окно написания процедуры

Окно написания процедуры содержит 3 основных части:

1. **Object** – список объектов, с которыми связана процедура. Список аналогичен списку выбора объектов панели **Properties**.
2. **Procedure** – список событий, с которыми связывается процедура. События отсортированы в алфавитном порядке.
3. Окно ввода.

*При написании любой процедуры следует контролировать, верно ли выбраны объект и его событие.*

**Некоторые события формы**

**Load** Форма загружается, но в момент возникновения данного события ни один объект формы не создан, поэтому невозможно задать свойства объектов формы. Можно открыть таблицы базы данных и ввести команды настройки.

**Init** Инициализация формы. Объекты формы уже созданы и можно задать какие-либо начальные значения.

**Refresh** Обновление формы.

**Release** Закрытие формы.

**Click** На форме выполнен щелчок левой кнопкой мыши.

**Некоторые события текстового поля**

**Click** Одинарный щелчок левой кнопкой мыши по полю.

**DblClick** Двойнойщелчок левой кнопкой мыши по полю.

**GotFocus** Фокус (курсор) входит в поле.

**LostFocus** Фокус(курсор) покидает поле.

**Некоторые события раскрывающегося списка**

**Valid** Событие, наступающее при выборе значения из списка.

### ЗАДАНИЕ 14. Осуществление простых вычислений на форме

Большое количество вычислений будет произведено на одной из самых важных форм проекта – форме, с помощью которой будет оформляться выдача дисков в прокат и формирование договора проката дисков. Кроме наличия вычислений, данная форма сама по себе будет иметь достаточно сложную структуру.

1. Запустите проект **dvd.pjx**.
2. С помощью мастера форм создайте форму **oformlenie.scx**, вид которой представлен на рис. 60:
   1. обратите внимание, что в основной части формы отображаются данные таблицы **prokat**, поэтому она будет указана в качестве родительской в мастере форм;
   2. обратите внимание, что в табличной части Grid отображаются данные таблицы **sostav**, поэтому она будет указана в качестве подчиненной в мастере форм.

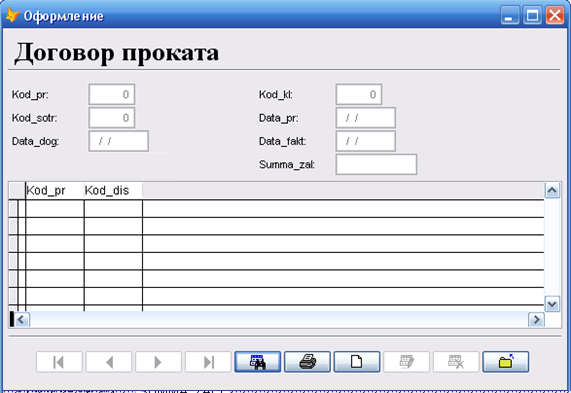


Рис. 60 – Первоначальный вид формы **oformlenie**

1. Отредактируйте форму так, чтобы она прияла вид, представленный на рис. 61:

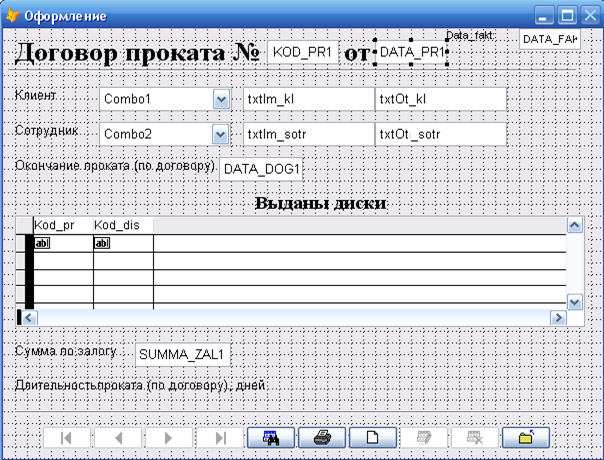


Рис. 61 – Структура преобразованной формы

* 1. создание раскрывающихся списков Клиент (**Combo1**) и Сотрудник (**Combo2**) выполните аналогично действиям ЗАДАНИЯ 8 или ЗАДАНИЯ 9 (при раскрытии списков должны отображаться фамилия, имя и отчество клиента или сотрудника соответственно);
  2. добавление полей, отображающих имена и отчества клиентов и сотрудников, выполните перетаскиванием из окна **DataEnvironment** или аналогично п.9 ЗАДАНИЯ 7;
  3. скройте поле **data\_fakt1** и соответствующую надпись, используя свойство **visible**;
  4. выполните изменение расположения и шрифта полей и надписей.

1. С помощью созданной формы добавьте в базу данных две записи о договорах проката. *Обратите внимание:*
   1. *не забудьте использовать кнопки* *,* *,*  *для добавления, сохранения или редактирования записей;*
   2. *в окне запроса (рис. 58) следует выбрать первый переключатель, поскольку добавление будет выполнено только в основную часть формы, а не в* ***Grid****;*
   3. *номера договоров проставляются автоматически, т.к. соответствующее поле таблицы базы данных имеет тип* ***Integer(AutoInc)****;*
   4. *формат вводимых дат –* ***мм/дд/гггг*** *(скорее всего, данный формат не удобен для пользователя, в дальнейшем он будет изменен);*
   5. *при выборе фамилий в раскрывающихся списков имена и отчеств меняются на соответствующие не сразу, а только после нажатия кнопки формы****Сохранить*** *или при щелчке в полях (этот недостаток в дальнейшем также будет устранен).*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Договор №** | **Начало проката** | **Клиент** | **Сотрудник** | **Окончание проката** |
| 1 | 08/13/2014 | Иванов Иван Иванович | Андреев Андрей Андреевич | 08/20/2014 |
| 2 | 08/30/2014 | Петров Петр Петрович | Маркина Ирина Сергеевна | 09/10/2014 |

1. Изменим формат даты на привычный для пользователя формат **дд.мм.гггг**. Это германский формат даты. Формат даты может быть изменен при помощи команды **SETDATE формат**. Необходимо определиться, когда формат должен быть изменен. Формат должен быть установлен при загрузке формы или при обновлении окна формы, поэтому данную команду следует связать с событиями **load** или **refresh** формы. Для этого выполните следующие действия:
   1. выполните двойной щелчок левой кнопкой мыши по свободной области формы, в результате чего на экране откроется окно ввода кода (рис. 62); убедитесь, что в списке **Object** отображается имя формы, а в списке **Procedure** – событие **Load**.
   2. в окне ввода разместите команду изменения формата даты (рис. 62); *обратите внимание на особенности ввода: после набора части* ***set*** *и нажатия пробела появляется список доступных параметров, где можно выбрать следующую часть* ***date*** *двойным щелчком левой кнопки мыши (можно после нажатия пробела ввести первую букву* ***d*** *и в списке автоматически будет выделен первый параметр, начинающийся на эту букву), аналогичные действия происходят после ввода части* ***date*** *– это особенности технологии I****ntelliSense*** *в FoxPro, которая прощает ввод команд пользователя.*

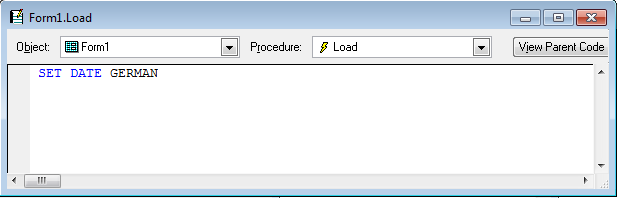


Рис. 62 – Изменение формата даты при загрузке формы

1. Запустите форму и убедитесь, что формат даты в полях изменен.
2. Добавьте на форму вычисляемое поле, в котором будет рассчитываться срок проката в днях:
   1. на панели инструментов **FormControls** выберите объект **TextBox** и выполните в свободном месте формы рядом с соответствующей надписью щелчок левой кнопкой мыши – обратите внимание, что на форме появилось поле **Text1** (номер поля может отличаться, учтите это при выполнении последующих пунктов задания);
   2. теперь необходимо написать строку программы, которая позволит рассчитывать количество дней проката; прежде всего следует верно выбрать объект и событие, при наступлении которого будет выполняться вычисление. В качестве такого события можно выбрать событие **refresh** формы – т.е. при изменении окна формы будет происходить пересчет количества дней. Выполните двойной щелчок на свободном месте формы, в результате чего откроется окно ввода кода процедуры (рис. 62). Убедитесь, что в списке **Object** отображается **Form1**, а в списке **Procedure** выберите событие **refresh**. Введите следующие программные строки (рис. 63):

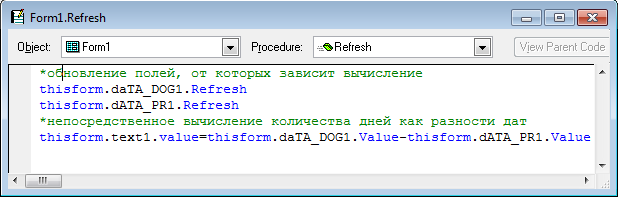


Рис. 63 – Вычисление количества дней проката дисков

1. Запустите форму, выполните переходы по записям и убедитесь в правильности расчета длительности проката.
2. Отредактируйте форму так, чтобы при выборе фамилии клиента динамически менялись соответствующие имя и отчество (*обратите внимание, что на данный момент изменение происходит при сохранении записи формы или при щелчке в этих полях*). Поскольку изменение имени и отчества зависит от выбора фамилии, то, очевидно, что соответствующая процедура будет связана именно с раскрывающимся списком фамилий клиентов **Combo1**. Изменение происходит после осуществления выбора, поэтому процедуру следует связать с событием **valid**. Выполните двойной щелчок на раскрывающемся списке, в результате чего будет запущено окно ввода кода процедуры, в поле **Obiect** которого отображается **Combo1** (список фамилий клиентов). В поле **Procedure** выберите событие valid и введите команду **thisform.refresh** – обновление формы, в т.ч. и полей с именем и фамилией.
3. Действуя аналогично, отредактируйте форму так, чтобы при выборе фамилии сотрудника динамически менялись соответствующие имя и отчество.
4. Проверьте правильность выполнения задания, изменив в договоре **№2** клиента на **Сергеев Сергей Сергеевич**.
5. Завершите работу с проектом.

### ЗАДАНИЕ 15. Программное изменение данных столбца Grid

В результате выполнения задания при оформлении выдачи в прокат каждого диска автоматически будет проставляться отметка **Выдан** (поле **Disk.v\_prok**принимает значение **.T.**).

1. Запустите проект **dvd.pjx**.
2. Откройте для редактирования в конструкторе форму **oformlenie.scx**.
3. Прежде всего, отредактируйте Grid так, чтобы она приняла вид, представленный на рис. 64 – в таком варианте при оформлении каждого проката дисков есть возможность фиксировать, какие именно диски выданы на руки клиенту, а так же можно рассчитать стоимость проката каждого диска. *Обратите внимание, что* ***Grid*** *будет содержать поля двух таблиц базы данных, поэтому для настройки* ***Grid*** *следует использовать только панель* ***Properties****, а не построитель* ***Builder***.

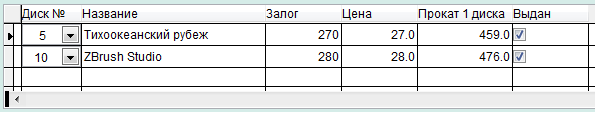


Рис. 64 – **Grid** после редактирования

1. добавьте в **DataEnvironment** таблицу **Disk** (рис. 65);
2. добавьте в Grid еще пять столбцов (ЗАДАНИЕ 11, п.15);
3. измените их заголовки на **Название**, **Залог**, **Цена**, **Прокат 1 диска**, **Выдан** соответственно (ЗАДАНИЕ 11, п.10);
4. свяжите столбец **Название** с полем **disk.nazv**, **Залог** – с полем **disk.zalog**, **Выдан** – с полем **disk.v\_prok** (ЗАДАНИЕ 11, п.16);
5. столбец **Цена** сделайте вычисляемым по формуле **disk.zalog\*0.1** (ЗАДАНИЕ 13);
6. столбец **Прокат 1 диска** сделайте вычисляемым по формуле **disk.zalog+0.1\*disk.zalog\*(thisform.daTA\_DOG1.Value-thisform.dATA\_PR1.Value)** (ЗАДАНИЕ 13);
7. измените тип управляющего элемента в столбце **Выдан** на **CheckBox** (ЗАДАНИЕ 12, п.3 – п.7).

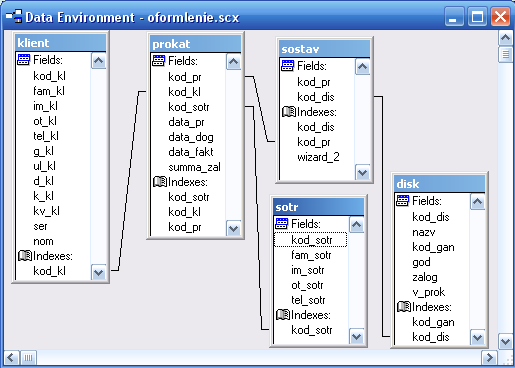


Рис. 65 – Окно **DataEnvironment** для формы **oformlenie**

1. Измените тип управляющего элемента столбца **kod\_dis** на **ComboBox**. В этом списке будут выбираться номера дисков, выдаваемых в прокат.
   1. перейдите в режим редактирования Grid (**Edit**);
   2. выберите на панели **Form Controls** инструмент **ComboBox** и выполните щелчок в столбце **kod\_dis** в любом месте под заголовком (обратите внимание, что в строке выбора объектов в панели **Properties** появился объект **Combo1**);
   3. выберите в панели **Properties** столбец с добавленным элементом (**Column2**) и измените его свойства: **CurrentControl** – **Combo1**, **Sparse** – **.F.**, *обратите внимание, что изменился элемент в ячейке столбца*;
   4. выберите в панели Properties добавленный объект Combo1 и измените его свойства:
      1. **RowSoursType** – 6-Fieldes
      2. **RowSource** –disk,kod\_dis
      3. **BoundColumn** –
      4. **BoundTo**– .T.
      5. **ControlSource** – sostav.kod\_dis
2. Измените заголовок столбца **kod\_dis** на **Диск №**.
3. Скройте столбец **kod\_pr** (ЗАДАНИЕ 11, п.12).
4. Запустите созданную форму и отредактируйте первый договор, оформив выдачу в прокат диска №5:
   1. нажмите кнопку  и в окне запроса параметров ввода выберите **Add record to child (grid) only**;
   2. в столбце **Диск №** в раскрывающемся списке выберите №5;
   3. нажмите кнопку .
5. Повторяя действия п.7, отредактируйте первый договор, оформив выдачу в прокат диска №10.
6. Вероятно, в Grid полностью не отображается номер диска из-за выравнивания в столбце и высоты строки. Чтобы устранить эти недостатки:
   1. перейдите в режим редактирования Grid (ЗАДАНИЕ 11, п.4);
   2. выделите столбец Диск № (ЗАДАНИЕ 11, п. 5);
   3. на панели **Properties** измените значение свойства **Alignment** на **2-MiddleCenter**;
   4. увеличьте высоту строки подобно изменению ширины столбцов (ЗАДАНИЕ 11, п.8-в).
7. Разработайте процедуру, которая позволит при выборе каждого диска, выдаваемого в прокат, проставлять отметку в столбце **Выдан**. Прежде всего, следует определиться с тем, с каким объектом и каким событием будет связана соответствующая процедура. Отметка ставится, при выборе каждого диска, т.е. процедуру следует связать со списком в столбце **Диск №** и с его событием **valid**. Выполните следующие действия:
   1. выберите объект Combo1 в столбце Диск №, используя панель Properties (рис. 66);

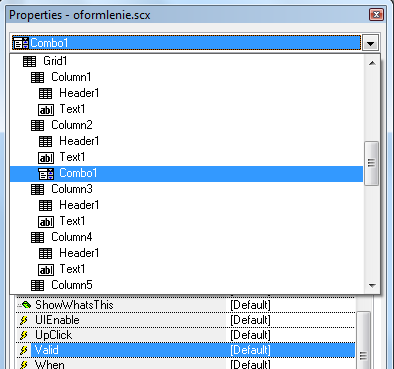


Рис. 66 – Выбор объекта в панели **Properties**

* 1. в панели **Properties** перейдите на вкладку **Methods**, выберите событие **Valid** (рис. 66) и выполните на нем двойной щелчок левой кнопкой мыши, в результате чего на экране откроется окно ввода кода процедуры (рис. 67);
  2. введите процедуру:

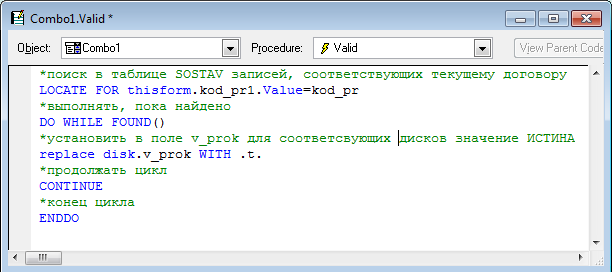


Рис. 67 – Окно ввода процедуры

1. Запустите форму, перейдите в режим редактирования записей и выберите диск №5 повторно. *Обратите внимание, что все диски в табличной части* ***Grid*** *стали помечены, как* ***выданные***.
2. Завершите работу с проектом.

### ЗАДАНИЕ 16. Вычисление суммы данных столбца Gridна форме

Подсчет итоговых сумм по столбцам **Grid** на форме выполняется при помощи процедуры, содержащей циклическую конструкцию. В результате выполнения задания будет подсчитана сумма по столбцам**Прокат 1 диска** – итоговая сумма проката.

Следует определиться с тем, с каким объектом и каким событием будет связана соответствующая процедура. Итоговая сумма меняется (накапливается) при выборе каждого диска, т.е. процедуру следует связать со списком в столбце **Диск №** и с его событием **valid**.

Итак, итоговая сумма будет меняться каждый раз при выборе диска. При просмотре данных с помощью формы **oformlenie** всегда будет отображаться значение, полученной при последнем выборе дисков на прокат. Чтобы устранить этот недостаток, итоговые суммы обязательно надо сохранять в поле таблицы базы данных. Поэтому сначала следует уяснить, какое поле таблицы базы данных будет использовано для подсчета итоговой суммы проката дисков.

В процессе получения третьей нормальной формы модели базы данных (п. 2.2.6) была приведена расчетная формула (**5**) для нахождения итоговой стоимости проката:

**Сумма**=**0.1**\***Сумма\_залога**\*(**дата\_оконч\_дог–дата\_проката**)+**сумма\_залога**, (5)

**Дата\_оконч\_дог** и **дата\_проката** отображаются в полях формы, а **сумма\_залога** – поле, хранящееся в таблице базы данных. Поэтому можно найти именно итоговую сумму по столбцу **Залог**, а затем воспользоваться формулой (**5**).

В результате выполнения задания будет получена форма, структура которой представлена на рис. 68.

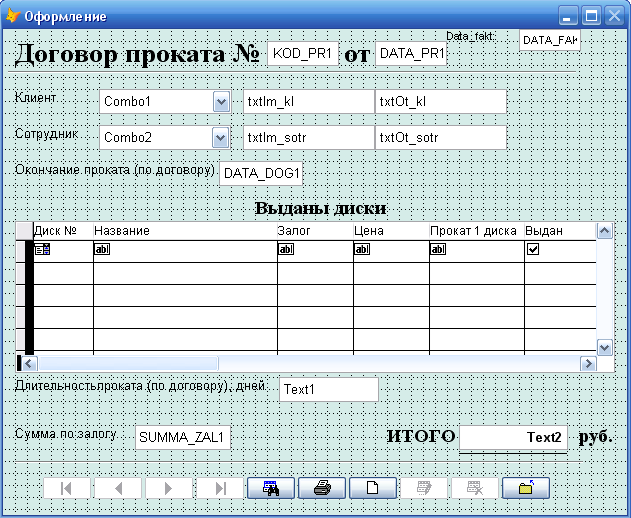


Рис. 68 – Структура формы **ofofrmlenie**

1. Запустите проект **dvd.pjx**.
2. Откройте для редактирования в конструкторе форму **oformlenie.scx**.
3. Подсчет итоговой суммы будет выполняться при наступлении события **valid** раскрывающегося списка в столбце Диск №, для которого уже написана процедура при выполнении ЗАДАНИЯ 15. Существует возможность отобразить список всех процедур, связанных с формой. Для этого, находясь в режиме конструктора формы (форма **оformlenie** как раз открыта в режиме конструктора) выполните команду меню **Tools** – **DocumentView**, в результате чего на экране отобразится окно, представленное на рис. 69. Если оно пустое, то выполните щелчок левой кнопкой мыши в любом месте формы.

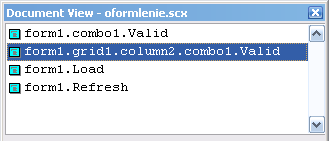


Рис. 69 – Окно **DocumentView**

*Обратите внимание, что здесь приведены все объекты и соответствующие события, для которых на форме есть хоть какие-то процедуры.*

1. В окне **DocumentView** выполните щелчок левой кнопкой мыши на объекте**form1.grid1.column2.combo1.valid**, в результате чего отобразится окно ввода кода процедуры (рис. 67).
2. Для расчета итоговой суммы по столбцу **Залог** отредактируйте имеющуюся процедуру, добавив новые программные строки (рис. 70):

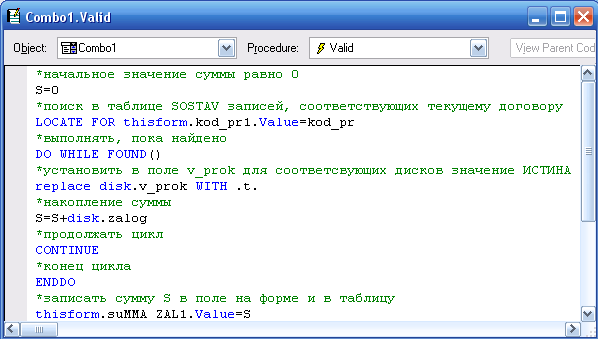


Рис. 70 – Процедура для события valid столбца Диск №

1. Запустите форму, перейдите в режим редактирования записей и выберите диск №5 повторно. *Обратите внимание, что в поле сумма по залогу рассчитана сумма данных в столбце* ***Залог***.
2. Отредактируйте договор №2, оформив выдачу в прокат диска № 2 (ЗАДАНИЕ 15, п.7). Обратите внимание на изменения в столбце **Выдан** и в поле **Сумма по залогу**.
3. Повторно откройте форму для редактирования.
4. Создайте на форме вычисляемое поле, в котором будет рассчитана стоимость проката дисков по формуле (**5**), повторяя действия ЗАДАНИЯ 14, п.7. В окне ввода процедуры для события refresh формы добавьте строки (рис. 71): (*обратите внимание на запись двух последних строк – по сути это одна программная строка, но разделенная на две с помощью знака точки с запятой – такой способ используется в тех случаях, когда команда не помещается в одной строке; по сути эти две строки лучше ввести в одну*)

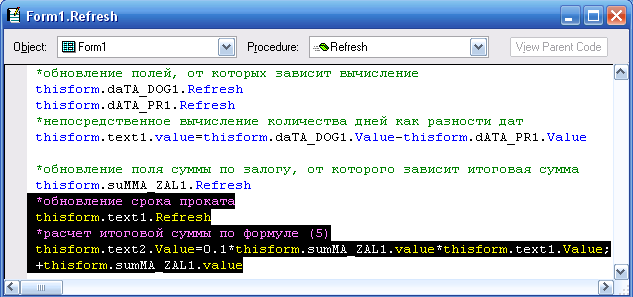


Рис. 71 – Обновленная процедура события refresh формы

1. Запустите форму. Отредактируйте договор №2, добавив в выдачу диск №3. Обратите внимание, что итоговая сумма – сумма значений столбца **Прокат 1 диска**. *Обратите внимание, что при выборе диска итоговая сумма не меняется, а расчет происходит только при нажатии на кнопку* ***Saverecord****.* Если требуется, чтобы расчет происходил непосредственно при выборе диска, то отредактируйте процедуру (рис. 70), добавив в нее выделенные строки из процедуры события **refresh** формы (рис. 71).
2. Приведите форму к виду, представленному на рис. 72:
   1. поля и надписи, присутствующие в конструкторе формы (рис. 68), но отсутствующие при просмотре (рис. 72), сделайте скрытыми, используя свойство **visible**;
   2. белый цвет фона полей можно установить, используя свойство
   3. выполните добавление надписей (удобно копировать существующие надписи и менять для копий свойство **Caption**);
   4. для отображения значений в столбце Цена без десятичных знаков измените свойства:
      1. столбца (**Column**) – **Format** = 99999[[3]](#footnote-4), **InputMask** = 99999;
      2. управляющего элемента в этом столбце **Text1** – **Format** = 99999, **InputMask** = 99999;
   5. аналогичным образом можно настроить отображение значений в столбце **Прокат 1 диска**;
   6. формат отображения значения в поле ИТОГО (text2) можно изменить, используя те же свойства **Format** и **InputMask**, установив в них значение 999999.99[[4]](#footnote-5).

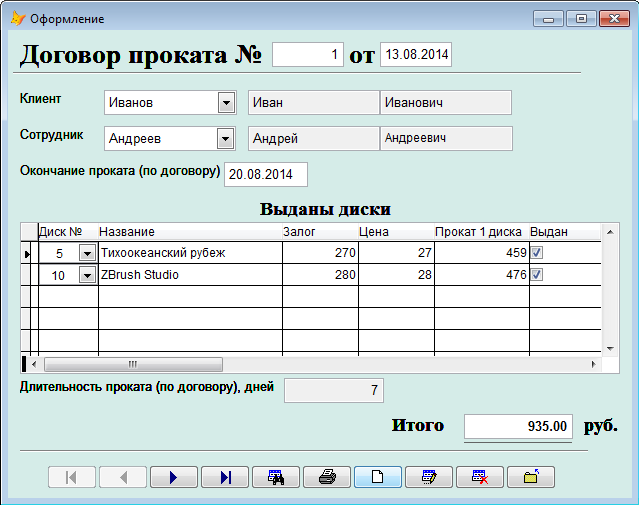


Рис. 72 – Итоговый вариант формы **oformlenie**

1. Проверьте работу формы.
2. Завершите работу с проектом.

### ЗАДАНИЕ 17. Программное управление объектами формы

Итак, уже разработана форма, позволяющая оформлять выдачу дисков в прокат. Теперь необходимо разработать форму, с помощью которой будет оформляться возврат дисков, а также редактирование договора проката. Новая форма позволит снять с возвращенных дисков отметку о выдаче и провести перерасчет стоимости при возврате дисков раньше или позже указанного в договоре срока.

1. Запустите проект **dvd.pjx**.
2. Создайте в проекте копию формы **oformlenie**:
   1. в папке **Dvd\Forms** выделите файлы **oformlenie.scx** и **oformlenie.sct**и создайте их копии в той же папке, присвойте копиями имена **redakt.scx** и **redakt.sct** соответственно;
   2. в окне проекта выделите объект Forms, нажмите кнопку Add и в появляющемся окне отыщите форму **redakt.scx**и нажмите кнопку ОК.

*Обратите внимание, что в проекте в группе Forms появилась форма* ***redakt.***

1. Отредактируйте форму так, чтобы ее структура приняла вид (рис. 73):

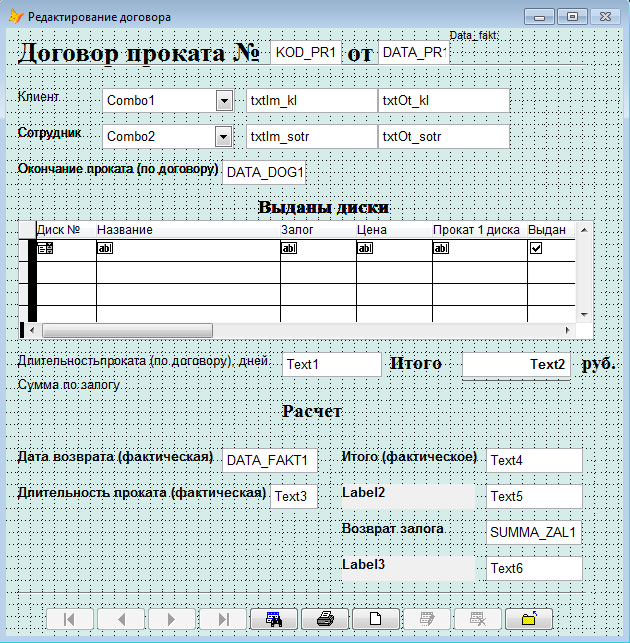


Рис. 73 – Структура формы **redakt**

*Обратите внимание:*

* *часть надписей (****Дата возврата (фактическая)****,* ***Итого (фактическое)****,* ***Длительность проката (фактическая)****,* ***Возврат залога****) получены копированием надписей, уже существующих на форме; об этом свидетельствует цвет их фона;*
* *поля* ***Data\_fact1****,* ***Summa\_zal1*** *– уже присутствовали на форме, но были скрыты; теперь они отображены;*
* *надписи* ***Label2*** *и* ***Label3*** *– особенные, их содержимое будет меняться в зависимости от того, возвращены ли диски в срок, досрочно или позже срока; они добавлены вручную с панели* ***FormControls****;* ***имена надписей могут отличаться, это следует учитывать при написании процедуры****;*
* *поля* ***Text3****,* ***Text4****,* ***Text5****,* ***Text6*** *– добавлены вручную с помощью панели* ***FormControls****; в них будут рассчитаны данные по возвращении дисков,* ***имена текстовых полей могут отличаться, это следует учитывать при написании процедуры****.*

1. Теперь следует определиться с алгоритмом, который будет использован для расчета и реализован в виде процедуры.

Если диски возвращены позже положенного срока, то:

* произошла просрочка дисков (*Label2, Text5*);
* фактическая стоимость проката дисков больше суммы по договору;
* клиенту возвращается сумма залога, но клиент платит за просрочку по договору, и в этом случае либо прокат остается должен клиенту, либо клиент – прокату (*Label3, Text6*).

Если диски возвращены раньше положенного срока, то:

* произошел досрочный возврат дисков (*Label2, Text5*);
* фактическая стоимость проката дисков меньше суммы по договору;
* клиенту возвращается сумма залога и разница за досрочный возврат дисков , и в этом случае прокат остается должен клиенту (*Label3, Text6*).

Если диски возвращены в установленный срок (*Label2, Text5*), то клиенту возвращается сумма залога (*Label3, Text6*).

1. Реализуйте данный алгоритм в виде процедуры для события **refresh** формы (введите текст процедуры ниже уже имеющегося кода, используя возможности технологии IntelliSense, копирование и вставку элементов строк и не обращая внимание на регистр символов):

\*обновление фактической даты

thisform.daTA\_FAKT1.Refresh

\*расчет фактической длительности

thisform.text3.value=thisform.daTA\_FAKT1.Value-thisform.dATA\_PR1.Value

\*расчет итого (фактического) *(присваивание лучше написать в одну строчку)*

thisform.text4.Value=0.1\*thisform.sumMA\_ZAL1.value\*thisform.text3.Value;

+thisform.sumMA\_ZAL1.value

\*определение: просрочка, досрочно или в срок

IF thisform.daTA\_FAKT1.Value>thisform.dATA\_dog1.Value

thisform.label2.Caption='Просрочка'

thisform.text5.Value=thisform.text4.Value-thisform.text2.Value

IF thisform.text5.Value>thisform.suMMA\_ZAL1.Value

thisform.label3.Caption='Долгзаклиентом'

thisform.text6.Value=thisform.text5.Value-thisform.suMMA\_ZAL1.Value

ELSE

thisform.label3.Caption='Возвратклиенту'

thisform.text6.Value=thisform.suMMA\_ZAL1.Value-thisform.text5.Value

ENDIF

ELSE

IF thisform.daTA\_FAKT1.Value<thisform.dATA\_dog1.Value

thisform.label2.Caption='Досрочныйвозврат'

thisform.text5.Value=thisform.text2.Value-thisform.text4.Value

thisform.label3.Caption='Возвратклиенту'

thisform.text6.Value=thisform.suMMA\_ZAL1.Value+thisform.text5.Value

ELSE

thisform.label2.Caption='Возвратвсрок'

thisform.text5.visible= .F.

thisform.label3.Caption='Возвратклиенту'

thisform.text6.Value=thisform.suMMA\_ZAL1.Value

ENDIF

ENDIF

1. Запустите форму на выполнение и убедитесь в правильности работы процедуры:
   1. отредактируйте договор №1, изменив фактическую дату возврата на 18.08.2014, сохраните изменения; обратите внимание на рассчитанные данные;
   2. отредактируйте договор №1, изменив фактическую дату возврата на 21.08.2014, сохраните изменения; обратите внимание на рассчитанные данные;
   3. отредактируйте договор №1, изменив фактическую дату возврата на 31.08.2014, сохраните изменения; обратите внимание на рассчитанные данные;
   4. отредактируйте договор №1, изменив фактическую дату возврата на 20.08.2014, сохраните изменения; обратите внимание на рассчитанные данные.
2. Теперь следует настроить снятие отметки о выдаче дисков при их возврате. Соответствующий флажок в Grid должен выключаться после заполнения поля фактической даты возврата, поэтому возможно связать соответствующую процедуру с событием **LostFocus** поля **DataFact1**. Перейдите в окно написания кода и введите процедуру (рис. 74):

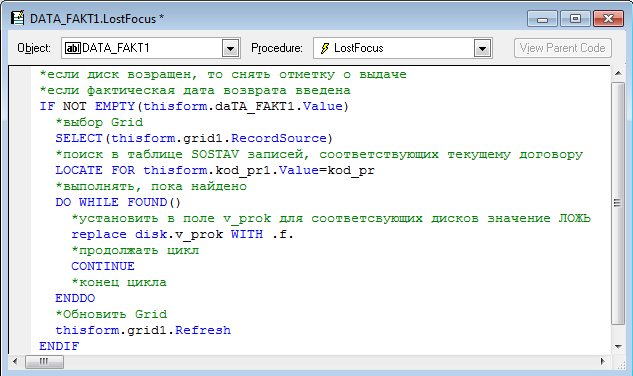


Рис. 74 – Процедура для снятия отметки о выдаче для возвращенных дисков

1. Запустите форму на выполнение, перейдите в режим **редактирования** договора №1, установите курсор в поле фактической даты возврата и нажмите кнопку **Add/Saverecord**. Обратите внимание, что флажок **Выдан** снят.
2. Перейдите к договору №2, убедитесь в том, что здесь отметка о выдаче сохранена.
3. При необходимости измените фон добавленных надписей, используя свойство **BackColor**.
4. При необходимости настройте формат вывода денежных сумм в рассчитанных полях так, чтоб отображалось два знака после десятичной точки (ЗАДАНИЕ 16, п.11-е).
5. Завершите работу с проектом.

## 4.6. Размещение на форме объекта CommandButton

Объект **CommandButton** (Кнопка) применяется для управления пользователем определенной операцией или последовательностью операций. Кнопки размещаются на формах с целью запуска определенных процедур.

**Некоторые свойства CommandButton**

**Caption** содержание надписи на кнопке

**FontCharSet** шрифт, начертание и размер надписи на кнопке

**FontUnderline** подчеркивание надписи на кнопке

**FontBold** полужирное начертание надписи на кнопке

**Alignment** выравнивание надписи внутри кнопки

**FontName** шрифт надписи

**BackColor** цвет кнопки

**Visible** видимость кнопки

**Некоторые свойства Command Button**

**Click** нажатие на кнопку

### ЗАДАНИЕ 18. Разработка простейших форм поиска данных

В результате выполнения задания будут разработаны формы поиска диска по названию и году выпуска. Их можно условно отнести к простейшим, т.к. результатом поиска станет таблица Browse, а сами формы будет построены на основе одной таблицы. Подобный поиск уже был реализован при выполнении ЗАДАНИЯ 4, но теперь будут разработаны диалоговые окнадля ввода части имени диска или года его выпуска.

1. Запустите проект **dvd.pjx**.
2. Создайте форму в режиме конструктора (ЗАДАНИЕ 7).
3. С помощью панели FormControls разместите на форме элементы: **Label**, **TextBox**, **CommandButton** так, чтобы она приняла вид, представленный на рис. 75.

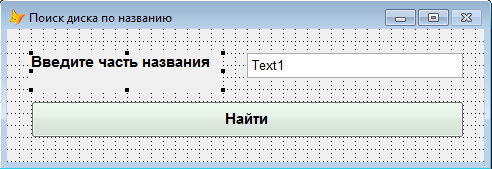


Рис. 75 – Структура формы поиска диска по названию

1. Теперь необходимо разработать саму процедуру поиска, которая будет запускаться после ввода части названия диска и нажатия на кнопку**Найти**. Процедуру следует связать с событием **Click** кнопки. Выполните двойной щелчок левой кнопкой мыши на кнопке**Найти**, в результате чего будет открыто окно ввода процедуры. Убедитесь, что в списке **Object** отображается кнопка **Command1**, а в списке **Procedure** – событие **Click**. Введите следующую процедуру:

\*обновление параметра поиска

thisform.text1.refresh

\*сохранить часть названия диска в переменную, удалив пробелы

x=ALLTRIM(thisform.text1.Value)[[5]](#footnote-6)

\*поиск в поле nazv без пробелов

LOCATE FOR ATCC(x,ALLTRIM(nazv))<>0[[6]](#footnote-7)

\*выполнить, если найдено

IF FOUND()

\*вывести, если найдено

BROWSE FIELDS nazv:H='Название', v\_prok:H='Наличие' FOR ATCC(x,ALLTRIM(nazv))<>0

ELSE

MESSAGEBOX('Диски не найдены',64,'Сообщение')

ENDIF

1. Добавьте в источник данных формы **DataEnvironment** таблицу, в которой проводится поиск и из которой выводятся данные – **Disk.dbf**.
2. Сохраните форму под именем **poisk\_nazv.scx**.
3. Запустите форму на выполнение. В качестве параметра поиска введите любой набор символов, например **абвгд**. Нажмите кнопку**Найти**. Убедитесь, что в результат на экране отображается сообщение об отсутствии дисков.
4. Введите в качестве параметра поиска **колон**. Нажмите кнопку**Найти**. Обратите внимание, что ширина столбца Название слишком велика, а в столбце наличие отображается буква **F** – значение «ложь» для поля **v\_prok** логического типа. Отредактируйте строку с командой **Browse** следующим образом (*текст команды должен быть записан в одной строке*):

BROWSE FIELDS nazv:H='Название':100, nal=IIF(v\_prok,'нет','да')[[7]](#footnote-8):H='Наличие' FOR ATCC(x,ALLTRIM(nazv))<>0

1. Повторите поиск и убедитесь в том, что изменились ширина столбца **Название** и содержимое столбца **Наличие**.
2. Создайте еще одну форму **Poisk\_god.scx** для поиска диска по году выпуска (рис. 76):

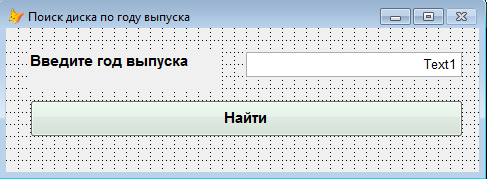


Рис. 76 – Структура формы поиска дисков по году выпуска

1. Обратите внимание на разницу в полях **Text1** на созданных формах: в первом случае для имени используется выравнивание по левому краю, а во втором – по правому. Это вызвано различием в типах данных: в первой форме поле **Text1** относится к строковому типу данных (**Character**), а во второй – к числовому (**Numeric**).
2. Измените типа поля **Text1** в форме **Poisk\_god**. Для этого в контекстном меню поля выберите команду **Builder**и в появляющемся окне **Text Box Builder** на вкладке **Format** в списке **Data Type** выберите тип данных **Numeric**.
3. Скопируйте уже разработанную процедуру поиска из формы **poisk\_nazv** и примените ее на форме **poisk\_god**:

\*обновление параметра поиска

thisform.text1.refresh

\*сохранить часть названия диска в переменную, удалив пробелы

x=thisform.text1.Value

\*поиск в поле nazv без пробелов

LOCATEFORgod>=x

\*выполнить, если найдено

IF FOUND()

\*вывести, если найдено

BROWSEFIELDSnazv:H='Название':100,god:H=’Год’, nal=IIF(v\_prok,'нет','да') :H='Наличие' FORgod>=x

ELSE

MESSAGEBOX('Диски не найдены',64,'Сообщение')

ENDIF

1. Добавьте в источник данных формы **DataEnvironment** таблицу, в которой проводится поиск и из которой выводятся данные – **Disk.dbf**.
2. Запустите созданную форму и проведите поиск дисков, выпущенные позже 2010 года. Убедитесь в правильности поиска.
3. Завершите работу с проектом.

# 5. РАЗРАБОТКА ОТЧЕТОВ

## 5.1. Общие сведения об отчетах. Рекомендации по созданию отчетов

**Отчет** – это особая форма представления данных, предназначенная для вывода на печать.

Отчет позволяет сгруппировать записи таблиц баз данных(по нескольким уровням); провести вычисления на основе полей таблиц баз данных; провести итоговые вычисления по полям таблиц баз данных.

Отчеты могут быть созданы на основе одной или нескольких таблиц БД.

Отчет, создаваемый в Visual FoxPro, может быть представлен в табличном виде или в свободной форме.

**Табличный отчет** — это напечатанная таблица, в которой строка представляет собой запись, а каждый из элементов строки содержит поле исходной таблицы или вычисляемое поле. Данные в таблице упорядочены. Табличные отчеты используются для печати данных, представленных в виде списка.

При подготовке писем, почтовых этикеток, договоров, квитанций поля используемых в отчете таблиц должны располагаться в специально выделенных для них местах. В этом случае табличный отчет не подходит и используются отчеты в **свободной форме**. При создании такого отчета вы можете воспользоваться стандартным форматом, автоматически создаваемым Visual FoxPro для каждой таблицы, в котором поля исходной таблицы располагаются вертикально. Используя конструктор отчетов, вы можете разрабатывать собственные форматы отчета, где поля исходной таблицы будут расположены там, где вам нужно.

**Рекомендации по созданию отчета**

Если вам требуется создать простой табличный отчет, в котором используются поля из одной таблицы, то трудностей, скорее всего, у вас не возникнет. При разработке более сложного отчета, прежде чем приступить к его созданию, вы должны четко представлять, что же хотите получить в результате. Поэтому продумайте приведенные ниже вопросы.

* С какой целью создается настоящий отчет?
* Отчет какого вида вы создаете (табличный, в свободной форме или наклейки)?
* Информация из каких таблиц должна быть представлена в отчете?
* Из используемых в отчете таблиц, какая таблица будет главной, а какая — подчиненной?
* По каким параметрам будет упорядочиваться информация в отчете?
* Будет ли при создании отчета использоваться группировка данных и по каким параметрам?

Получив четкие ответы на поставленные вопросы, вы облегчите свою работу при создании отчета.

Создавая первые отчеты, прежде чем приступить к работе с конструктором отчетов, прорисуйте формат отчета на бумаге. Это избавит вас от ошибок и сократит время, затрачиваемое на разработку отчета.

**Структура отчета**

Отчет подразделяется на области. Области отчета дают возможность определить, где и когда отображать данные на странице отчета или этикетке. Предположим, что вы создаете каталог дисков в DVD-прокате. Для того, чтобы систематизировать информацию в отчете, можно использовать его различные области. Приведем пример информации, которую можно разместить в различных областях отчета:

* Название каталога. Оно будет отображаться в области заголовка отчета, который появляется один раз в начале отчета (**Title**).
* Общее количество дисков в DVD-прокате, используя область итогов, которая расположена в конце отчета (**Summary**).
* Название отчета, номер страницы и текущую дату, которые будут располагаться в верхнем или нижнем колонтитуле страниц, соответственно на каждой странице (**PageHeader**, **PageFooter**).
* Линии, отделяющие диски одного жанра и одной категории, которые появляются каждый раз, когда меняется жанр и категория (**GroupHeader**, **GroupFooter**).

Так же можно включать во все области отчета различные объекты: поля, надписи, вычисляемы поля и т.д.

При создании отчеты уже содержат в себе несколько областей по умолчанию, однако можно добавить и другие области в отчет.

Таблица 15. Области отчета

| **Область отчета** | **Описание** |
| --- | --- |
| **Title**  Заголовок | Отображает информацию о заголовке отчета или его титульном листе. Необязательная часть, может быть размещена на отдельной странице в начале отчета или на первой странице непосредственно перед областью верхнего колонтитула. Не может быть длиннее одной страницы |
| **PageHeader**  Верхний колонтитул страницы | Отображает информацию, которая представлена один раз в начале каждой страницы. Обязательная область на всех страницах, кроме случая, когда области заголовка (Title) и итогов (Summary) расположены на отдельных страницах. Не может быть длиннее одной страницы. |
| **GroupHeader**  Верхний колонтитул группы | Отображает информацию или данные, которые представлены один раз на группу, предшествует группе. Добавляется автоматически для каждой группы данных, созданных пользователем. |
| **Detail**  Детализация | Это основная часть отчета, здесь отображается содержимое таблиц БД. Одна область детализации обязательна в отчете. Пользователь может по желанию добавить еще одну область детализации. Область может быть длиннее одной страницы. |
| **GroupFooter**  Нижний колонтитул группы | Добавляется автоматически для каждой группы, созданной пользователем, и содержит вычисляемые данные для группы данных. |
| **PageFooter**  Нижний колонтитул страницы | Отображает информацию, которая представлена один раз в конце каждой страницы. Обязательная область на всех страницах, кроме случая, когда области заголовка (Title) и итогов (Summary) расположены на отдельных страницах. Не может быть длиннее одной страницы. |
| **Summary**  Итоговые данные | Необязательная часть, может быть размещена на отдельной странице в конце отчета или непосредственно перед нижним колонтитулом страницы на последнем листе отчета. Содержит итоги по всему отчету. |

**Источник данных отчета**

Так же, как и форма, отчет имеет окно источника данных, на основе которых он построен (**Data Environment**). Сюда можно добавлять таблицы БД, запросы и временные таблицы (курсоры). Из окна Data Environment можно добавлять поля в отчет обычным перетаскиванием, как и для формы.

**Средства создания отчетов**

В Visual FoxPro для создания отчетов можно использовать следующие средства.

**Report Wizard** (**Мастер отчета**). Позволяет достаточно быстро создать отчет, применяя сортировку, группировку данных и заданный вами стиль оформления. Для создания готового отчета с помощью мастера достаточно ответить на ряд вопросов в диалоговых окнах мастера.

**Report Designer** (**Конструктор отчета**). В конструкторе отчетов вы можете разрабатывать собственные или модифицировать отчеты, созданные с помощью мастера.

При создании отчета можно использовать любые средства, предлагаемые Visual FoxPro, какие вам кажутся более удобными. Конечно, больше возможностей предоставляет конструктор отчетов, позволяющий создавать отчеты как в табличном виде, так и в свободной форме.

Наиболее удобным вариантом является создание отчета с помощью мастера и последующая настройка его с помощью конструктора.

## 5.2. Создание простых отчетов с помощью мастера

При разработке приложений не следует злоупотреблять простейшими отчетами, поскольку большая часть из них не информативна. Например, DVD-прокате вряд ли полезно иметь печатный документ, отображающий сведения только о категориях дисков или только о жанрах. Гораздо полезнее иметь отчет, который содержит сведения обо всех дисках в прокате, с группировкой их по жанрам и категориям, или же отчет с перечнем дисков в наличии.

### ЗАДАНИЕ 19. Создание простейшего отчета с помощью мастера

1. Запустите проект **dvd.pjx**.
2. С помощью мастера отчетов создайте отчет, содержащий сведения о клиентах проката **Klient.frx**. Для этого перейдите на вкладку **Documents (Docs)**, выделите объект **Reports** и нажмите кнопку **New**. В появляющемся окне запроса способа создания выберите **ReportWizard**. В результате будет запущен мастер форм. Выберите тип мастера – **ReportWizard** (создание отчета на основе одной таблицы).
3. Выполните шаги мастера:
   1. Выбор полей: выбранные здесь поля будут отображены в отчете. Используя кнопки  или , добавьте в область **SelectedFields** все поля таблицы **Klient** (рис. 77).

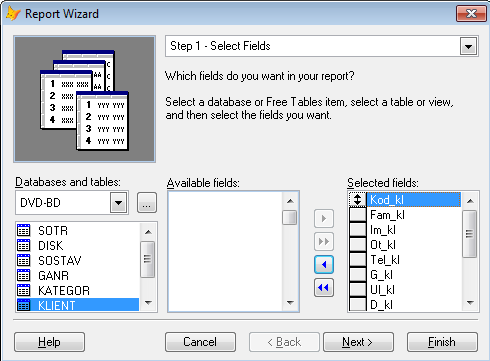


Рис.77 – Мастер отчетов – шаг 1

* 1. Группировка записей: позволяет выбрать признак группировки и провести итоговые вычисления (рис. 78), пропустите этот шаг, нажав кнопку **Next**;

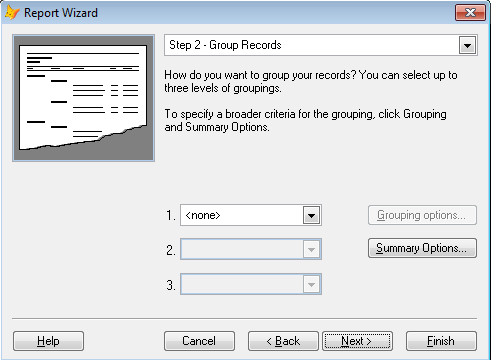


Рис. 78 – Мастер отчетов – шаг 2

* 1. Выбор стиля отчета: внешний вид отчета (рис. 79); выберите вариант **Ledger**;

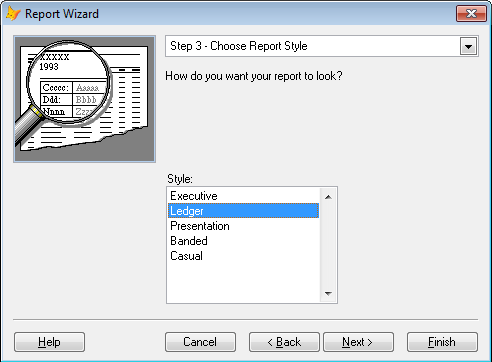


Рис.79 – Мастер отчетов – шаг 3

* 1. Макет отчета: количество столбцов в отображении полей, ориентация бумаги, вариант расположения полей в столбец или в строку; оставьте установки, как на рис. 80:

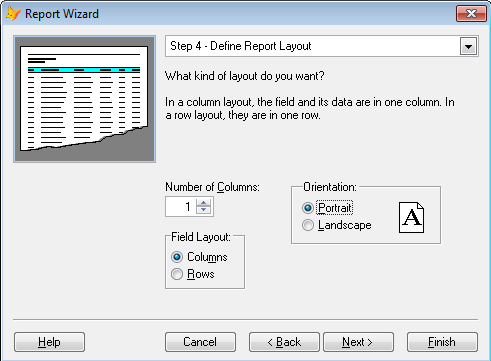


Рис. 80 – Мастер отчетов – шаг 4

* 1. Сортировка записей: можно указать поле, с сортировкой по которому записи будут отображаться в полях отчета, а также порядок сортировки – по возрастанию (**Ascending**) или убыванию (**Descending**); установите поля для сортировки в соответствии с рис. 81:

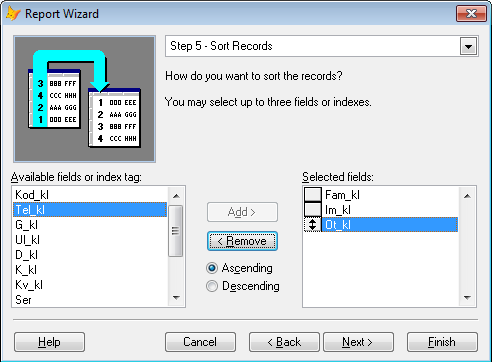


Рис.81 – Мастер отчетов – шаг 5

* 1. Завершение (рис. 82):
     1. Заголовок отчета: заголовок будущего печатного документа. По умолчанию совпадает с названием таблицы, для которой создается отчет.
     2. Варианты сохранения (сохранить и запустить, просто сохранить, сохранить и открыть конструктор).
     3. Предпросмотротчета.

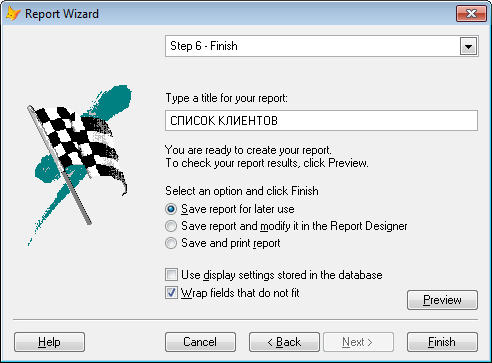


Рис.82 – Мастер отчетов – шаг 6

* 1. Сохраните созданныйотчет под именем **Klient** в папке **D:\DVD\Reports**. *Обратите внимание, что в папке появились два фала:* ***Klient.frx*** *и* ***Klient.frt****.*

1. Обратите внимание, что в менеджере проекта в группе **Reports** появился созданныйотчет.
2. Выполните просмотр отчета, выделив его в менеджере проекта и нажав кнопку **Preview** (рис. 83).

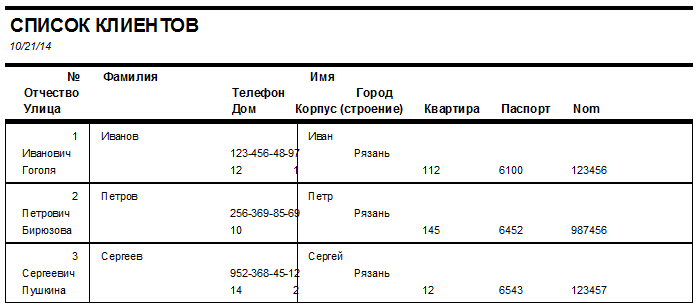


Рис. 83 – Отчет в режиме просмотра

1. Обратите внимание, что на экране появилась панель **PrintPreview** (рис. 84), которая позволяет переходить между страницами многостраничного отчета, переходить на первую и последнюю страницы отчета, переходить к странице с указанным номером, изменять масштаб просмотра отчета, выйти из просмотра отчета и вывести его на печать.



Рис. 84 – Панель Print Preview

1. Завершите просмотр отчета.
2. Откройте отчет в режиме конструктора, для чего выделите отчет в менеджере проекта и нажмите кнопку **Modify**, в результате чего на экране отобразится макет отчета (рис. 85).

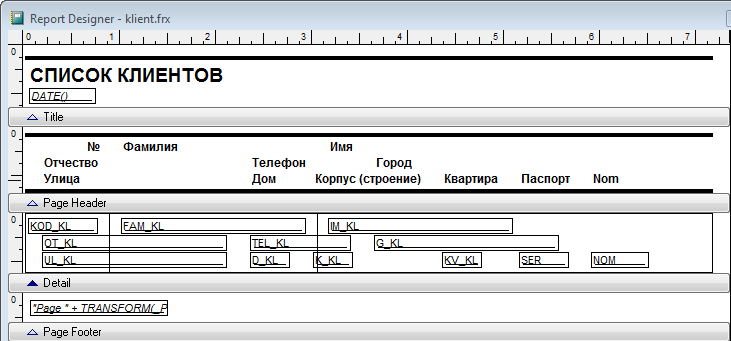


Рис. 85 – Отчет в режиме конструктора

1. Обратите внимание, что отчет содержит следующие области: **Title**, **PageHeader**, **Detail**, **PageFooter**.
2. Закройте отчет.
3. Завершите работу с проектом.

## 5.3. Создание отчетов с группировкой с помощью мастера

Группировка полезна в отчетах для таблиц с внешним ключом. Например, таблица **Ganr** имеет внешний ключ **kod\_kat** – т.е. несколько жанров относятся к одной категории. Поэтому целесообразно сгруппировать жанры по категориям, чтобы при отображении жанров одной категории наименование категории отображалось один раз.

В одном отчете может быть выполнено несколько группировок по нескольким признакам (уровни группировки). Например, жанры сгруппированы по категориям, а в каждом жанре содержится группа дисков.

В отчете может быть до 20 вложенных уровней группировки. После создания каждого уровня группировки в отчете появляется пара областей **GroupHeader** и **GroupFooter**.

**Возможности группировки:**

* Начинать каждую группу с отдельной страницы.
* Начинать нумерацию страниц с 1 в каждой группе.
* Печатать заголовок группы на каждой странице.

Группировку данных в отчете можно выполнить двумя способами:

* Мастер создания отчетов.
* Конструктор.

Мастер (шаг 2) позволяет выполнить группировку только при создании отчета на основе одной таблицы, локального представления (запроса) или курсора (временной таблицы). В то же время можно указать несколько уровней группировки.

Конструктор позволяет добавлять или удалять уровни группировки; настраивать параметры группы: каждая группа на новой странице, нумерация с 1 внутри каждой группы и т.д. Настройки выполняются в окне **ReportProperties** – вкладка **DataGrouping** (контекстное меню отчета – команда **Properties**).

Удобнее создавать отчеты с группировкой с помощью мастера.

Если предполагается, что отчет содержит данные нескольких таблиц и требует группировки, то исходные несколько таблиц следует объединить в локальное представление или курсор.

### ЗАДАНИЕ 20. Создание простого отчета с группировкой

1. Откройте проект **dvd.pjx**.
2. С помощью мастера на основе таблицы **Ganr** создайте отчет, содержащий перечень жанров дисков, сгруппированных по категориям дисков:
   1. на первом шаге мастера выберите все поля таблицы **Ganr**;
   2. на втором шаге мастера укажите поле для группировки – **kod\_kat** (рис. 86):

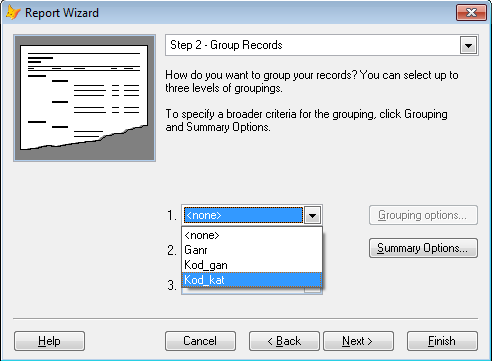


Рис. 86 – Выбор параметра группировки в мастере отчетов

* 1. на пятом шаге мастера установите сортировку по полю **Ganr**;
  2. на последнем шаге мастера введите заголовок отчета **Жанры и категории**;
  3. сохраните отчет под именем **Ganr.frx**.

1. Просмотрите созданный отчет. Он примет вид, представленный на рис. 87 (возможны незначительные расхождения в оформлении):



Рис. 87 – Отчет с группировкой

1. Завершите работу с проектом.

## 5.4. Создание отчетов с итоговыми вычислениями

Существует возможность провести в отчете итоговые вычисления среди значений поля по группе данных:

* по всем записям таблицы БД;
* по записям, отобранным с помощью запроса;
* по записям, сгруппированным по какому-либо признаку (в отчетах с группировкой).

Итоговые вычисления располагаются в специальной области отчета – **Summary**. Эта область и соответствующие вычисления может быть добавлена при использовании мастера или конструктора.

**Итоговые (агрегатные) функции:**

**Sum** Сумма значений указанного поля

**Avg** Среднее арифметическое значений указанного поля

**Count** Количество непустых значений в указанном поле

**Min** Минимальное среди значений указанного поля

**Max** Максимальное среди значений указанного поля

Мастер отчетов (шаг 2 – кнопка **Summary Options**) вызывает диалоговое окно **Summary Options**, в котором указаны доступные итоговые функции и поля таблицы базы данных, по которым можно провести итоговые вычисления. Пример окна приведен на рис. 88.

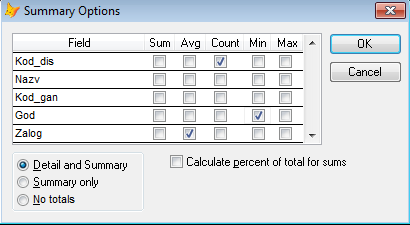


Рис. 88 – Окно настройки итоговых вычислений **Summary Options**

В окне **Summary Options** можно указать параметры отображения итоговых значений:

* **Detail and Summary** – в отчете отображается область данных, промежуточные итоговые значения по группировкам и конечные итоговые значения по отчету
* **Summary only** – в отчете отображается область данных, и конечные итоговые значения по отчету
* **No totals** – отображается только область данных, без итогов

Итоговые вычисления можно проводить не только в простых отчетах, но и в отчетах с группировкой. Именно в этом случае и будет использоваться параметр **Detail and Summary**

В отчете с итоговыми вычислениями появляется область **Summary**, которая содержит 2 объекта:

* Поле, которое содержит используемую итоговую функцию: **[TotalCount:]**, **[TotalSum:]**, **[Average:]** и т.д. Это поле по умолчанию отображается при просмотре отчета в оригинальном виде. Естественно, присутствие такой надписи в отчете при печати непонятно обычному пользователю. Но**это поле нельзя удалять из отчета**. Его просто можно сделать невидимым, используя цвет текста.
* Поле, которое указывает поле таблицы базы данных, по которому выполняются итоговые вычисления. Именно в нем и будет отображено итоговое значение при просмотре отчета.

Конструктор позволяет вручную добавить в отчет область **Summary** в отчет: контекстное меню отчета – команда **Properties** – вкладка **OptionalBands** – флажок **Report has Summary Band**.

Лучше создавать отчеты с итоговыми вычислениями с помощью мастера и редактировать с помощью конструктора.

### ЗАДАНИЕ 21. Создание простого отчета с итоговыми вычислениями

1. Запустите проект **dvd.pjx**.
2. С помощью мастера отчетов создайте отчет, содержащий сведения о клиентах проката **Klient-1.frx**(ЗАДАНИЕ 19), но на втором шаге нажмите кнопку SummaryOptions для отображения одноименного окна и укажите в нем создание поля с общим количеством клиентов проката (рис. 89). Остальные шаги мастера выполните аналогично ЗАДАНИЮ 19.

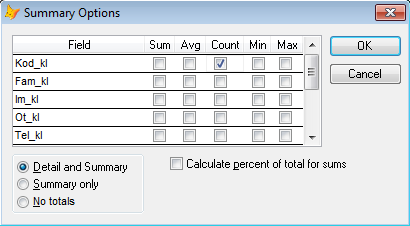


Рис. 89 – Настройки итоговых вычислений для количества записей

*Обратите внимание: в окне выбрана функция* ***Count*** *(количество) и поле первичного ключа, т.к. только это поле гарантированно не будет пустым в таблице* ***Klient****.*

1. Выполните просмотр созданного отчета и обратите внимание, что в отличие от отчета **Klient.frx** (рис. 83) в нижней части в нем появилось поле и общее количество клиентов проката (рис. 90).

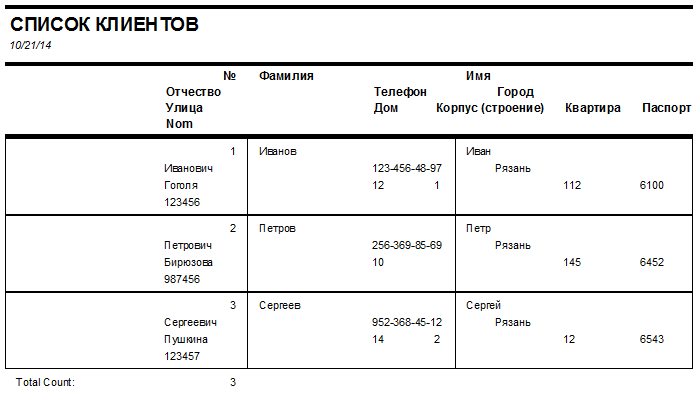


Рис.90 – Отчет с итоговым количеством

1. Завершите работу с проектом.

### ЗАДАНИЕ 22. Создание простого отчета с группировкой и итоговыми вычислениями

1. Запустите проект **dvd.pjx**.
2. С помощью мастера создайте отчет **Disk-1.frx** с группировкой (ЗАДАНИЕ 20) по жанрами с определением общего количества дисков и года выпуска самого «старого» диска:
   1. основа отчета – таблица **Disk**;
   2. поле для группировки – **kod\_gan**;
   3. **Summary Options** – **Max**дляполя**God**, **Count**дляполя**kod\_dis** (ЗАДАНИЕ 21);
   4. сортировка по названию диска;
   5. заголовок отчета – Диски и жанры.
3. Просмотрите созданный отчет (рис. 91, содержание полей может отличаться). Обратите внимание, что после каждой группы дисков, относящихся к одному жанру, появились поля **[CountFor:]** (и количество дисков одного жанра) и **[LowestFor:]** (и минимальный год выпуска дисков одного жанра).

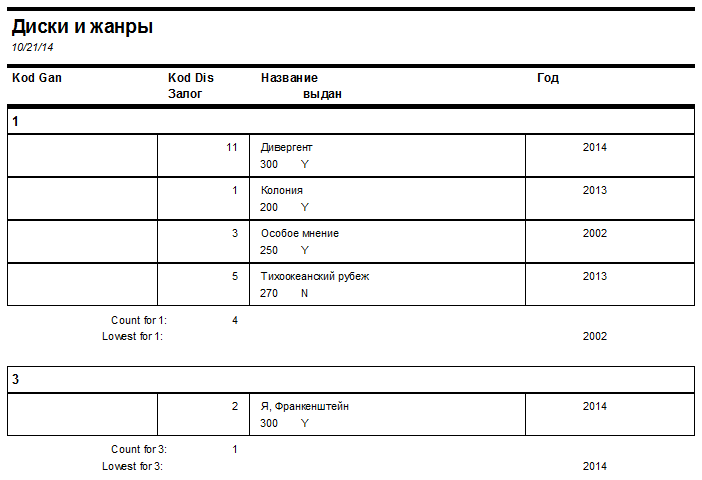


Рис.91 – Отчет с группировкой и итоговыми вычислениями по группе

1. Обратите внимание, что на панели **PrintPreview** стали активны кнопки перехода по страницам отчета. Используя кнопку , перейдите на последнюю страницу отчета и обратите внимание на то, что в конце страницы появились поля **[TotalCount:]** (и общее количество всех дисков проката) и **[Lowest:]** (и минимальный год выпуска из всех дисков). Таким образом, итоговые вычисления произведены как по каждой группе в отдельности, так и по всему отчету.
2. Завершите работу с проектом.

## 5.5. Использование конструктора для работы с отчетами

Вообще говоря, лучше не создавать отчеты с помощью конструктора, а только редактировать отчеты, уже созданные с помощью мастера.

Вызов конструктора отчета выполняется в менеджере проекта двойным щелчком левой кнопки мыши по названию отчета или нажатием кнопки **Modify**.

Именно в конструкторе отчета в явном виде отображены его области, а также основные элементы:

1. **Label** – Надпись;
2. **Field** – Поле;
3. **Line** – Линия;
4. **Rectangle** – Прямоугольник;
5. **RoundedRectangle** – Прямоугольник со скругленными углами;
6. **Picture** – Изображение.

Элементы отчета могут быть добавлены с помощью панели инструментов **Report Controls**: **View – Report Controls Toolbar**.



Рис.92 – Панель инструментов **ReportControls**

В режиме конструктора в отчет могут быть добавлены поля из окна **Data Environment**.

Для редактирования полей и надписей используется соответствующие окна: **Label Properties** и **Field Properties**, вызываемые двойным щелчкомлевой кнопкой мыши или командой **Properties** контекстного меню.

Свойства распределены по вкладкам, приведем некоторые из них.

Таблица 16. Некоторые свойства надписей и полей отчета

|  |  |
| --- | --- |
| **Надпись (Label)** | **Поле (Field)** |
| **Caption** – содержание надписи. | **Expression** – определение поля. Может быть полем таблицы БД или выражением, по которому вычисляется значение поля отчета |
| **Font** – настройки шрифта | |
| **Color** – настройки цвета. Для изменения цвета текста и цвета фона следует снять флажки использования цвета по умолчанию | |
| **BackStyle** – отображение или скрытие фона | |
|  | **FormatExpression** – формат отображения данных.  Например, XXX-XXX-XX-XX – телефон,  999,999.99 – формат отображения числа с 2-мя знаками после запятой.  *Некоторые настройки формата меняются в зависимости от типа данного.* |

### ЗАДАНИЕ 23. Использование конструктора для настройки внешнего вида отчета

1. Запустите проект **dvd.pjx**.
2. Откройте в конструкторе отчет **Klient-1.frx**, для чего выделите его в менеджере проекта и нажмите кнопку **Modify**. Обратите внимание, что отчет содержит 3 вида элементов: надписи, поля и линии.
3. Выделите надпись **Город**, выполните по ней двойной щелчок левой кнопкой мыши, в результате чего откроется окно **LabelProperties** (рис. 93). Измените свойство **Caption** на **г.**. Нажмите кнопку ОК.

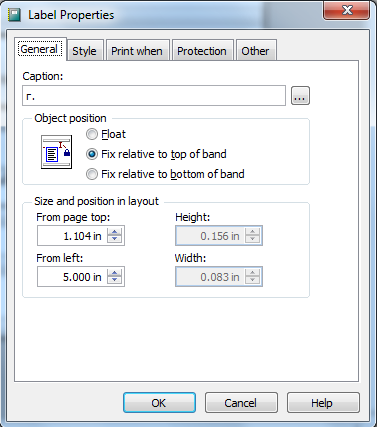


Рис.93 – Окно **Label Properties**

1. Действуя аналогично, измените содержание надписей **Улица** – на **ул.**, **Дом** – на **д.**, **Корпус(строение)** – на **корп. (стр.)**, **Квартира** – на **кв.**.
2. Выделите поле **DATE()** и перетащите его в правую часть области **Title**.
3. Измените размер поля **DATE()**, увеличив его по горизонтали, при помощи перетаскивания правого или левого маркера рамки выделения.
4. Установите отображение даты в формате **дд.мм.гггг**: отобразите для поля **DATE()** окно **FieldProperties** и выполните следующие установки (рис. 94):

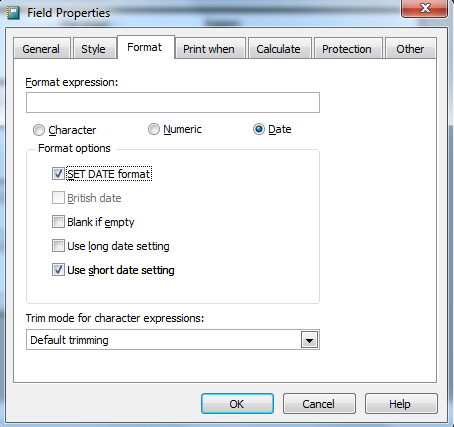


Рис.94 – Изменение формата даты

1. Выделите надпись **№** и нажмите клавишу **Delete** для ее удаления.
2. Удалите поле **kod\_kl**.
3. Удалите поле **tel\_kl**.
4. Добавьте это поле обратно в ту же область отчета:
   1. вызовите контекстное меню свободной части отчета (на сером фоне за пределом страницы) и выберите команду **DataEnvironment**; *обратите внимание, что в области источника данных отображается таблица* ***Klient****, на основе которой и построен отчет*;
   2. в таблице **Klient** выделите поле **tel\_kl** и перетащите его на прежнее место в отчете;
   3. оставьте только поле, а добавленную вместе с ним надпись **Телефон** удалите.
5. Наведите указатель мыши на название области отчета **Title**, чтобы он принял вид двунаправленной стрелки, и перетащите полосу (с заголовком **Title**) максимально вверх. *Обратите внимание, что ширина области заголовка уменьшилась. Аналогичным образом можно уменьшать и увеличивать ширину каждой области отчета.*
6. Выделите надпись заголовка **Список клиентов** и вызовите для нее окно **LabelProperties**. Перейдите на вкладку **Style** и установите флажок **Underline** для установки подчеркивания. Здесь же справа от поля **Font** нажмите кнопку  для изменения параметров шрифта. Установите размер шрифта равным **12**. Нажмите кнопку ОК.
7. Отобразите панель инструментов **ReportControls**, выбрав в меню **View**команду**Report Controls Toolbar**.
8. На панели инструментов нажмите кнопку **Label** и выполните щелчок левой кнопкой мыши внутри области отчета **PageHeader**. Обратите внимание, что появился мигающий текстовый курсор. Введите надпись **Адрес**. По окончании вода выполните щелчок в любом свободном месте отчета.
9. Выделите созданную надпись и, действуя аналогично п.9 установите размер шрифта равный **9** и начертание **полужирное**.
10. Обратите внимание на область **Detail**: она разделена вертикальными линиями на 3 столбца. Выделите правую или левую линии. Обратите внимание на то, что маркеры выделения появились вокруг прямоугольника (используя их, можно менять его размеры, но не выходя за границы области **Detail**; ее размер можно изменить подобно п.8). Выделите любую из двух оставшихся линий, скопируйте ее и вставьте обычным образом (через буфер обмена, используя контекстное меню или комбинации клавиш **Ctrl+C** и **Ctrl+V**).
11. Разместите копию линии так, чтобы она позволяла делить область **Detail** на 4 части. Для точного позиционирования линии (как и любого объекта отчета) можно использовать перетаскивание с помощью мыши, клавиши управления курсором, клавиши управления курсором при зажатой клавише **Ctrl**.
12. Вызовите контекстное меню отчета и выполните команду **PrintPreview** для просмотра отчета. Использование команды полезно для просмотра промежуточных результатов редактирования отчета.
13. Завершите просмотр отчета.
14. Обратите внимание на область **Summary** – ее содержимое нельзя менять, переносить в другие области отчета, можно изменить только формат отображения объектов:
    1. отобразите окно свойств поля **[TotalCount:]**; *содержимое поля* ***Expression*** *нельзя менять*;
    2. перейдите на вкладку **Style** (рис. 95), снимите флажок **Usedefaultforeground (pen) color**, с помощью кнопки  палитру цветов и выберите белый цвет; нажмите ОК. Обратите внимание, что поле стало невидимым, хотя и присутствует в отчете.

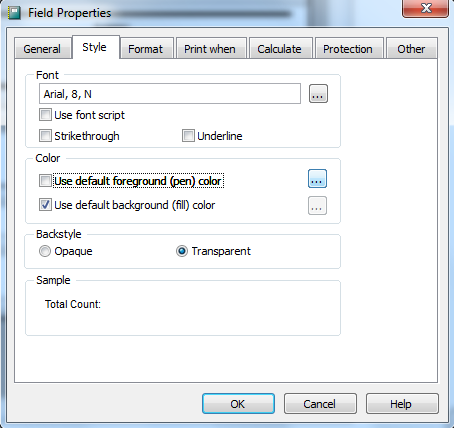


Рис.95 – Окно **Field Properties**

1. Настройте отчет так, чтобы он принял вид, представленный на рис. 96:*(при необходимости можете изменять размеры полей, областей отчета, а также менять расположение элементов отчета)*

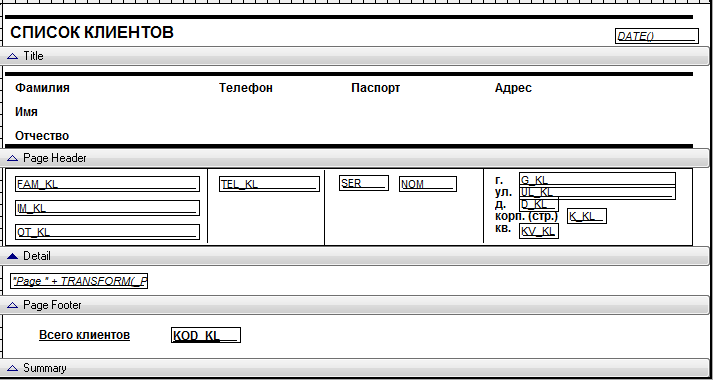


Рис.96 – Структура отчет **Klient-1**

1. Выполните просмотр отчета и убедитесь, что он принимает вид, представленный на рис. 97:

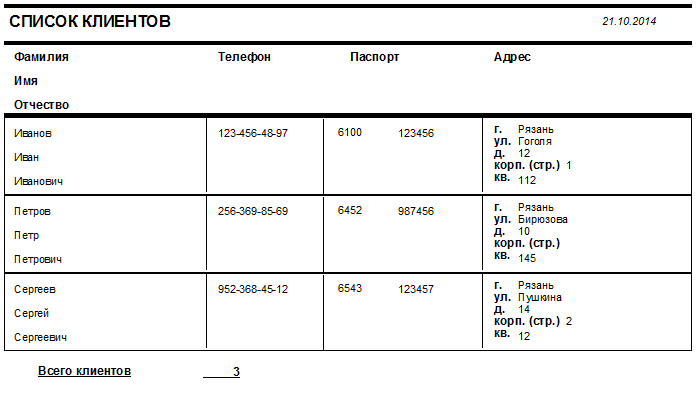


Рис. 97 – Итоговый вариант отчета **Klient-1**

1. Закройте отчет.

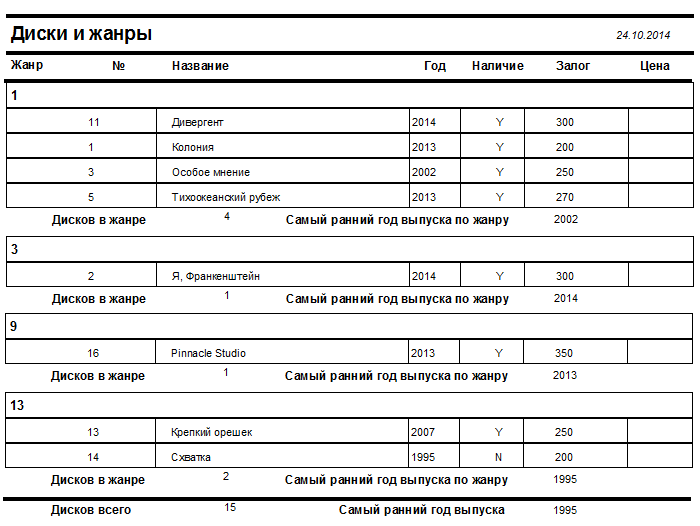


Рис. 98 – Отредактированный отчет **Disk-1.frx**

1. Действуя аналогично, приведите отчет **Disk-1.frx** к виду, представленному на рис. 98 (*для данных числового типа примените выравнивание по левому краю, используя флажок* ***LeftJustify*** *на вкладке* ***Format*** *окна* ***FieldProperties****; если данное не умещается в поле, то при просмотре представлено символами \*\*\*\* - в этом случае можно увеличить размер поля, расширяя рамку выделения или удалить шаблон отображения в поле* ***FormatExpression*** *вкладки* ***Format***).
2. Завершите работу с проектом.

## 5.6. Создание отчетов на основе нескольких таблиц

Мастер создания отчетов позволяет разрабатывать отчеты на основе максимум двух связанных таблиц баз данных (как и мастер форм). При этом создаваемый отчет внешне похож на подобную форму. При создании такого отчета (так же, как и при создании такой формы) следует определиться, какая из двух таблиц будет родительской, а какая – подчиненной. Это принципиально важно. Если в созданном отчете должны отображаться данные и из других таблиц баз данных, то они будут добавлены в отчет в режиме конструктора.

При создании отчета на основе двух таблиц, связанных отношением «один-ко-многим» создается отчет с группировкой, где в качестве поля группировки выступает ключ, за счет которого и реализуется связь между таблицами.

### ЗАДАНИЕ 24. Создание многотабличного отчета

Мастер создания отчетов позволяет разрабатывать отчеты на основе максимум двух связанных таблиц баз данных (как и мастер форм). При этом создаваемый отчет внешне похож на подобную форму. При создании такого отчета (так же, как и при создании такой формы) следует определиться, какая из двух таблиц будет родительской, а какая – подчиненной. Это принципиально важно. Если в созданном отчете должны отображаться данные и из других таблиц баз данных, то они будут добавлены в отчет в режиме конструктора.

При создании отчета на основе двух таблиц, связанных отношением «один-ко-многим» создается отчет с группировкой, где в качестве поля группировки выступает ключ, за счет которого и реализуется связь между таблицами.

В результате выполнения задания будет разработан начальный вариант печатного договора, выдаваемого на руки клиенту при оформлении выдачи дисков в прокат.

1. Запустите проект **dvd.pjx**.
2. Запустите мастер создания отчетов и выберите вариант **One-to-ManyReportWizard**для разработки отчета на основе таблиц **Sostav** и **Prokat** (в дальнейшем он будет доработан и приведен к виду, схожему с формой **oformlenie**). Следуйте шагам мастера:
   1. шаг 1 – выбор полей родительской таблицы, в данном случае, таблицы **Prokat** (рис 99): (*обратите внимание, что среди полей, добавленных в отчет, отсутствует поле* ***data\_fakt****, т.к. при оформлении договора фактическая дата возврата дисков не известна и не используется*)

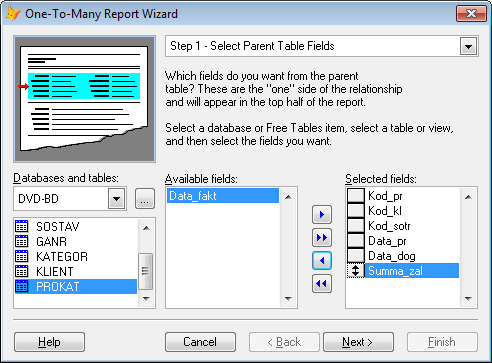


Рис.99 – Мастер создания отчета на основе двух таблиц, шаг 1

* 1. шаг 2 – выбор полей подчиненной таблицы, в данном случае, таблицы **Sostav** (рис. 100):

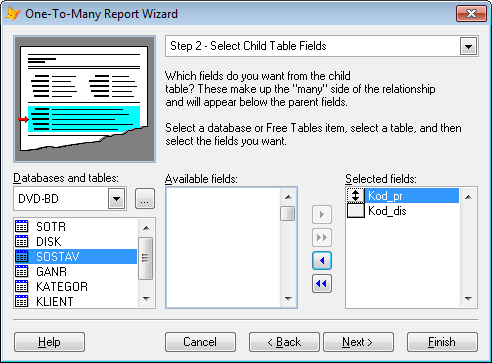


Рис.100 – Мастер создания отчета на основе двух таблиц, шаг 2

* 1. шаг 3 – выпор поля для связи таблиц, обычно определяется автоматически (рис. 101):

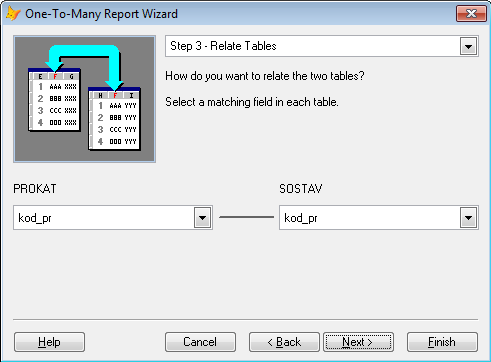


Рис.101 – Мастер создания отчета на основе двух таблиц, шаг 3

* 1. шаг 4 – определение параметров сортировки (рис. 81), сортировка в данном отчете не нужна;
  2. шаг 5 – выбор стиля отчета (рис. 102), при необходимости здесь можно указать необходимость проведения итоговых вычислений; в данном отчете итоговые вычисления не нужны;

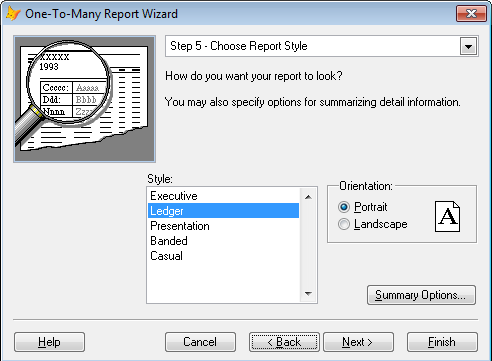


Рис.102 – Мастер создания отчета на основе двух таблиц, шаг 5

* 1. шаг 6 – завершение создания отчета (рис. 84); укажите заголовок отчета **Договор №** и сохраните отчет под именем **dogovor.frx**.

1. Выполните просмотр отчета и убедитесь, что они принял вид, представленный на рис. 103 (*содержимое полей может отличаться*). Обратите внимание, что в отчете отображается информация обо всех оформленных договорах.

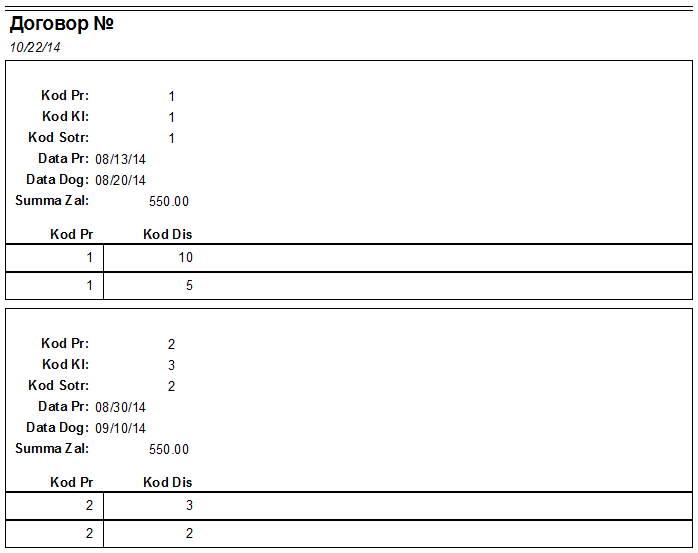


Рис. 103 – Отчет **dogovor.frx** в режиме просмотра

1. В режиме конструктора окно настройки группировки отчета, используя контекстно меню свободной области отчета → команда **Properties**→ вкладка**DataGrouping** или контекстно меню свободной области отчета → команда **DataGrouping** (рис. 104). Установите режим отображения каждой группы с новой страницы, используя переключатель **NewPage**.

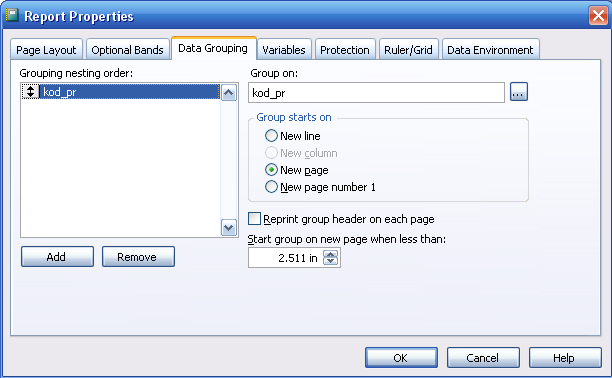


Рис. 104 – Окно настройки параметров группировки

1. Выполните просмотр отчета убедитесь, что каждая группа отображается на отдельной странице. Обратите внимание на панель **PrintPreview** (рис. 105), с помощью которой можно просматривать остальные страницы отчета.



Рис. 105 – Панель Print Preview

1. Поскольку создаваемый отчет по структуре должен быть похож на форму **oformlenie**, то следует добавить в него поля из других таблиц базы данных, а для этого нужно отредактировать источник данных отчета **DataEnvironment**. Приведите **DataEnvironment** к виду, представленному на рис. 106, добавив новые таблицы и изменив связи между ними. ***Обратите внимание: нельзя менять связь между исходными таблицами Prokat и Sostav.***[[8]](#footnote-9)

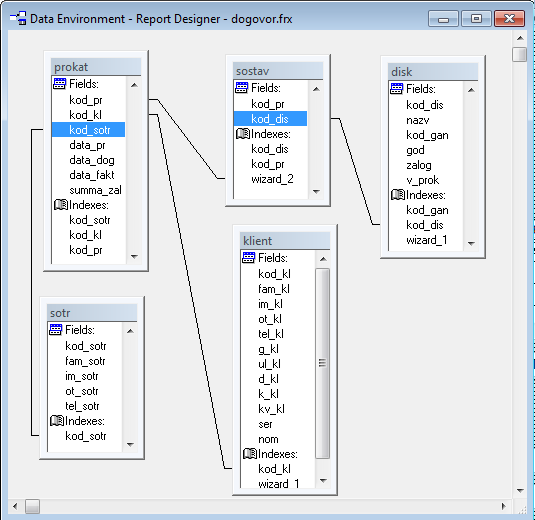


Рис. 106 –Источник данных отчета **dogovor**

1. Перетаскиванием из **DataEnvironment** добавьте недостающие поля в отчет и отредактируйте его таким образом (ЗАДАНИЕ 23), чтобы его структура приняла вид, представленный на рис. 107:
   1. *в процессе редактирования периодически выполняйте просмотр отчета командой контекстного меню* ***Print Preview***;
   2. обратите внимание, что в отчете отсутствуютполя ключей таблиц: **kod\_pr** из таблицы **Sostav**, **kod\_kl** из таблицы **Klient**, **kod\_sotr** из таблицы **Sotr** – ***не удаляйте их***, а используйте белый цвет шрифта;
   3. в отчет добавлены или изменены надписи;
   4. в отчете не используются области **Title** и **PageHeader**;
   5. для числовых полей установите выравнивание содержимого по левому краю, используя флажок **Leftjustify** вкладки **Format** диалогового окна **FieldProperties**;
   6. измените формат отображения дат (ЗАДАНИЕ 23, п.7).

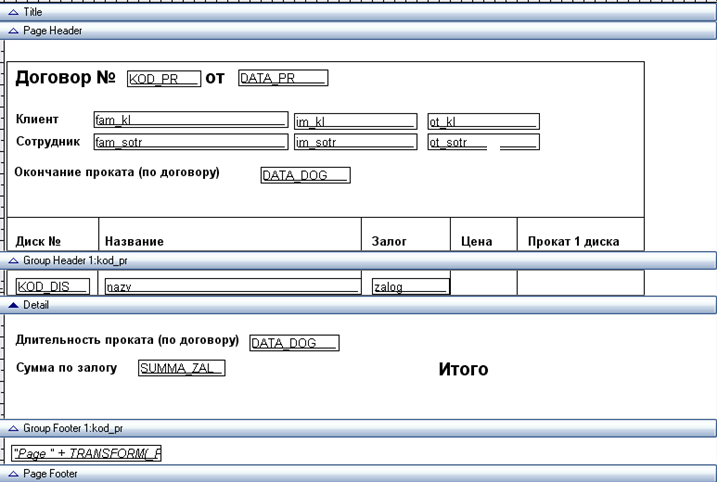


Рис. 107 – Структура отчета **Dogovor**

1. Выполните просмотр отчета.
2. Действуя аналогично, приведите отчет **Disk-1.frx** к виду, представленному на рис. 108.

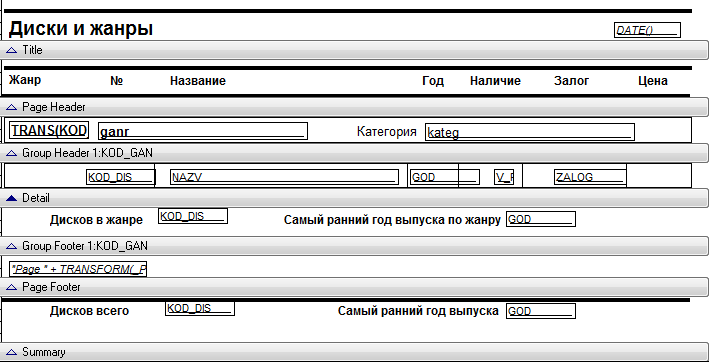


Рис. 108 – Макет отчета **Disk-1.frx**

1. Завершите работу с проектом.

## 5.7. Фильтрация данных отчета

При просмотре отчета, разработанного в ЗАДАНИИ 24, в нем отображается информация обо всех оформленных договорах проката дисков. В реальности этот документ должен выдаваться клиенту на руки непосредственно после оформления выдачи дисков. Представим себе ситуация – оформляется прокат дисков с номером 250, и требуется вывести на печать соответствующий договор. Что получится при использовании разработанного отчета? Отобразится документ из 250 листов, в котором надо сначала перейти к последней странице, а потом напечатать ее. Но это еще не самая сложная ситуация. Предположим, что при оформлении этого договора была допущена ошибка, и клиент ее заметил не сразу, а спустя несколько часов. Клиент возвращается в прокат, чтобы откорректировать и перепечатать договор, после которого, возможно, было оформлено достаточное количество выдач дисков в прокат. И в отчете придется искать страницу с номером 250. Конечно, существует возможность перейти к странице договора с заданным номером, но все же гораздо удобнее после оформления или редактирования проката нажатием одной кнопки «Печать» видеть на экране (и в дальнейшем распечатать) только один договор с нужным номером. Такую возможность предоставляет использование фильтрации в отчете.

Фильтрация возможно при наложении условия на отображаемые данные при помощи параметра **FOR условие**, где **условие** – выражение логического типа, и в отчете будут отображены только те записи, для которых условие верно.

### ЗАДАНИЕ 25. Привязка отчета к текущей записи формы

1. Запустите проект **dvd.pjx**.
2. Выполните просмотр отчета **dogovor.frx**. Закройте отчет. Обратите внимание на содержание окна Command (если оно не отображается на экране, то воспользуйтесь кнопкой  или командой меню **Window** → **CommandWindow**) – здесь появляется команда просмотра отчета **ReportForm** (рис. 109).

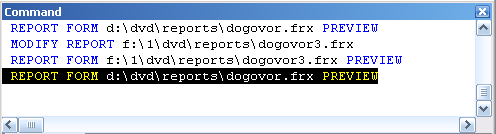


Рис. 109 – Команда просмотра отчета в окне **Command**

1. Установите курсор внутри строки с командой и нажмите клавишу **Enter**. Обратите внимание, что произошел запуск просмотра отчета.
2. Снова установите курсор внутри строки с командой и отредактируйте ее (рис. 110):

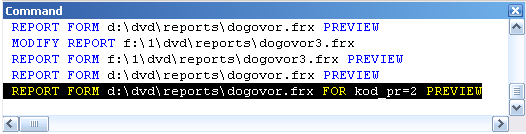


Рис. 110–Отредактированная команда просмотра отчета

1. Запустите отредактированную команду. Обратите внимание на содержимое появляющегося отчета – в нем отображается информация о договоре №2 – т.е. запись из таблицы **Prokat**, где **kod\_pr=2**. Это и есть применение фильтра к содержимому отчета.
2. Скопируйте последнюю выполненную команду в буфер обмена.
3. Откройте для редактирования форму **oformlenie**.
4. Выделите панель управляющих кнопок, вызовите на ней контекстное меню и выберите команду Edit, в результате чего вокруг панели появится голубая рамка ограничителя и кнопки станут доступны для редактирования по отдельности.
5. Выделите кнопку **Printreport** и выполните на ней двойной щелчок левой кнопкой мыши, что приведет к открытию окна ввода процедуры.
6. Убедитесь, что в окне процедуры отображается объект **cmdPrint** и событие **Click**. Вставьте из буфера обмена скопированную команду и отредактируйте процедуру, приведя ее у виду, представленному на рис. 111.

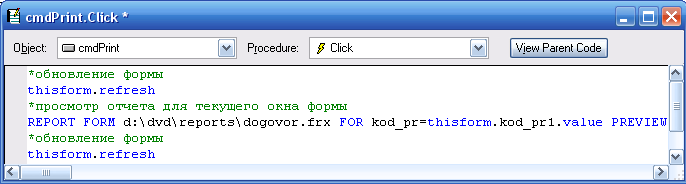


Рис. 111 – Вызов отчета для текущего окна формы

1. Запустите отредактированную форму и с ее помощью оформите договор проката дисков, на основе следующих данных:

№ договора 3

Клиент Петров Петр Петрович

Сотрудник Маркина Ирина Сергеевна

Начало проката 25.10.2014

Окончание проката 27.10.2014

Выданы диски №4, №15

1. Нажмите кнопку  и обратите внимание, что в появляющемся отчете представлены данные только по договору №3. Убедитесь, что данные верны.
2. Закройте отчет.
3. В форме перейдите на запись по договору №2 и нажмите кнопку . Убедитесь, что в появляющемся отчете представлены данные только по договору №2.
4. Скопируйте разработанную процедуру и добавьте ее к событию Click кнопки  формы redact.scx.
5. Завершите работу с проектом.

## 5.7. Размещение вычисляемых полей в отчете

Вычисляемые поля могут быть созданы в отчете при использовании режима конструктора. Выражения для вычислений могут быть введены для вновь создаваемого поля или для уже существующего поля отчета.

Для создания поля в отчете следует использовать инструмент **Field** панели **ReportControls**. Процедура добавления поля подобна добавлению поля на форме.

После выбора инструмента **Field** на панели **ReportControls** и размещения нового поля на форме автоматически откроется окно свойств поля **FieldProperties**(рис. 112). Если выражение для вычисления будет добавлено в уже существующее поле отчета, то следует отобразить окно свойств этого поля.

Написание выражения выполняется в поле **Expression**. При написании выражения могут быть использованы названия полей таблиц баз данных, от которых зависит будущее вычисляемое поле, и различные функции FoxPro.

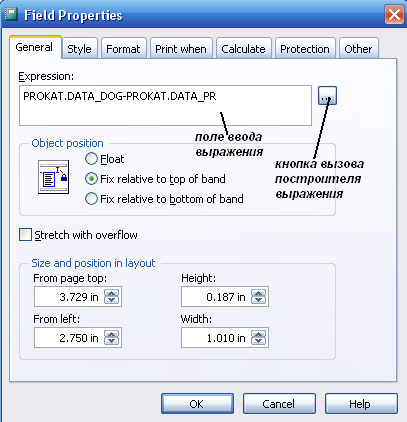


Рис. 112 – Окно свойств вычисляемого поля

Создание выражений может быть выполнено вручную, путем ввода в поле Expression, или создано с помощью построителя выражений **ExpressionBuilder** (рис. 113).

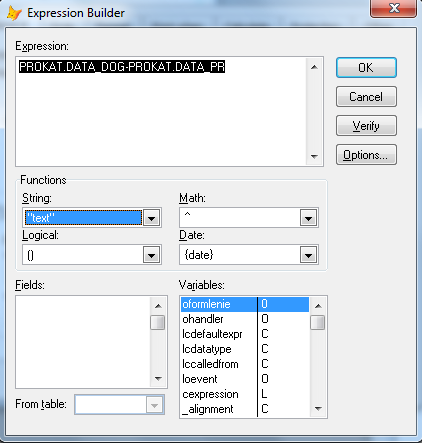


Рис. 113 – Окно построителя выражений **ExpressionBuilder**

Построитель предоставляет возможность использовать четыре группы функций. Выбор функций выполняется из раскрывающегося списка, в результате чего в поле ввода вставляется пустой шаблон функции, и пользователю остается просто добавить аргументы функций. Набор функций достаточно обширен, поэтому не предоставляется возможность изучить каждую из них. Но FoxPro имеет достаточно удобную справочную систему, где дается краткое описание функций и примеры их использования. Несмотря на то, что справка англоязычная, она достаточно понятна пользователю с базовыми знаниями языка.

Функции сгруппированы по категориям:

**String– текстовыефункции**

Приведем примеры функций:

ALLTRIM(expC), ALLT(expC) – убирает пробелы из выраженияexpC типа Character

LEFT(expC, N) – выбирает N символов из строки expC слева

+ - объединение строк

UPPER(expC) – преобразование строки expC к верхнему регистру смиволов

*Например*:

allt(Klient.F\_kl)+" "+allt(Klient.I\_kl)+" "+allt(Klient.O\_kl) – объединить поля с фамилией, именем, отчеством, установив между ними по одному пробелу

**Math – математические функции**

Приведем примеры функций:

Арифметические: +, –, \*, /, ^

VAL (expC) – преобразование строковой величины expC в числовую

ABS () – модуль числового значения

Тригонометрические функции

MAX() – нахождение максимального значения

MIN() – нахождение минимального значения

SUM () – суммирование значений

COUNT () – подсчет количества значений

AVG () – нахождение среднего арифметического значений

**Logical – логические функции**

Приведем примеры функций:

операции сравнения: <, >, >=, <=, <>, == операции сравнения

AND, NOT, OR – логические функции

EMPTY(exp) – проверяет, является ли аргумент exp пустым

IIF (условие, выражение1, выражение2) – если проверяемое условие истинно, то выполняется выражение1, иначе – выражение2

*Например*:

iif (disk.v\_prok,"выдан","в наличии")

**Date – функции работы с датами**

DATE() – получение текущей датаы

TTOD() – выделяет дату из аргумента типа DateTime

DAY() – выделяет день из аргумента типа Date

MONTH() – выделяет месяц из аргумента типа Date

YEAR() – выделяет год из аргумента типа Date

*Например*:

IIF(month(date())==month(PROKAT.DATA\_PR),1,2)

### ЗАДАНИЕ 26. Размещение вычисляемых полей в отчете

1. Запустите проект **dvd.pjx**.
2. Выполните просмотр отчета **disk-1.frx**. *Обратите внимание, что в отчете не отображается цена проката (т.к. она не хранится в базе данных, а вычисляется по формуле* ***(1): cena=0.1\*zalog****). Обратите внимание, что в столбце* ***Выдан*** *отображаются логические значения* ***F*** *и****Т****, не понятные, вообще говор, обычному пользователю)*.
3. Откройте отчет в конструкторе и создайте вычисляемое поле для нахождения цены проката:
   1. при необходимости отобразите панель **ReportControls** (Задание 23, п.14);
   2. выберите инструмент **Field** и выполните щелчок в нужном месте отчета в области **Detail**, в результате чего будет запущено окно **FieldProperties** (рис. 114);
   3. в поле **Expression** введите выражение для вычисления:

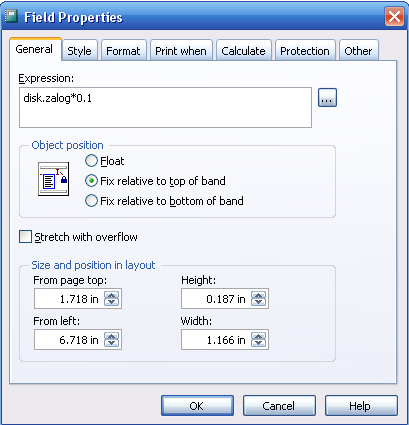


Рис. 114 – Создание вычисляемого поля

1. Выполните просмотр отчета и убедитесь в правильности вычислений.
2. Существует возможность не создавать вычисляемые поля «с нуля», а использовать поля, которые уже есть в отчете. В данном случае цена диска зависит от величины залога, поэтому можно использовать следующий вариант размещения вычисляемого поля:
   1. снова откройте отчет в конструкторе;
   2. выделите поле **zalog** и скопируйте его в буфер обмена (**Ctrl+C**или**Ctrl+Insert**);
   3. вставьте поле из буфера обмена (**Ctrl+V**или**Shift+Insert**), переместите его в область **Detail**;
   4. выполните двойной щелчок на добавленном поле для отображения окна **FieldProperties**;
   5. отредактируйте поле **Expression**, добавив умножение поля **zalog** на **0.1**.
3. Выполните просмотр отчета, убедитесь в правильности вычислений.[[9]](#footnote-10)
4. Оставьте в итоговом отчете одной из двух добавленных полей.
5. Отредактируйте поле **v\_prok** так, чтобы для дисков в наличии отображалось**Да**, для дисков в прокате отображалось **Нет**:
   1. вызовите окно **Field Properties**;
   2. вызовите построитель выражений **ExpressionBuilder**, используя кнопку ;
   3. скопируйте в буфер обмена поле **v\_prok**, присутствующее в поле **ExpressionforfieldonReport** (оно пригодится в новом выражении);
   4. в списке **Logical** найдите и выберите функцию **IIF()** (рис. 115), в результате чего она отобразится в области выражения;
   5. используя содержимое буфера обмена, отредактируйте выражение, чтобы оно приняло вид **IIF(V\_PROK,"Нет","Да")**.**[[10]](#footnote-11)**

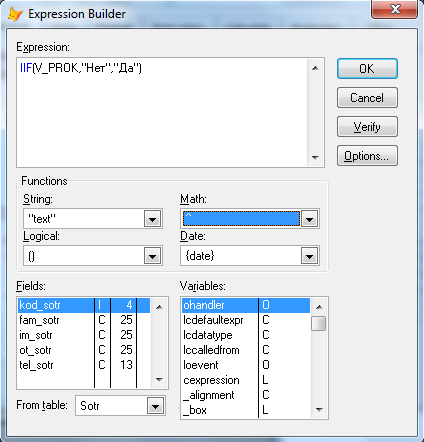


Рис. 115 – Создание вычисляемого поля с помощью функции FoxPro

1. Выполните просмотр отчета. Скорее всего в поле **v\_prok** отображается одна буква**Д** или **Н**. Чтобы устранить этот недостаток, в окне **Field Properties** на вкладке **Format** очистите поле **Format Expression**.
2. Просмотрите отчет повторно и убедитесь в правильности выполнения задания.
3. Откройте в конструкторе отчет **dogovor.frx**. Добавьте в него вычисляемые поля[[11]](#footnote-12) (рис. 116):
   1. Поле 1: **alltrim(klient.fam\_kl)+" "+alltrim(klient.im\_kl)+" "+alltrim(klient.ot\_kl)**
   2. Поле 2:**alltrim(sotr.fam\_sotr)+" "+alltrim(sotr.im\_sotr)+" "+alltrim(sotr.ot\_sotr)**
   3. Поле 2: **disk.zalog\*0.1**
   4. Поле 4:**disk.zalog\*0.1\*(PROKAT.DATA\_DOG-PROKAT.DATA\_PR)+disk.zalog**
   5. Поле 5:**PROKAT.SUMMA\_ZAL+(PROKAT.DATA\_DOG-PROKAT.DATA\_PR)\*PROKAT.SUMMA\_ZAL\*0.1**

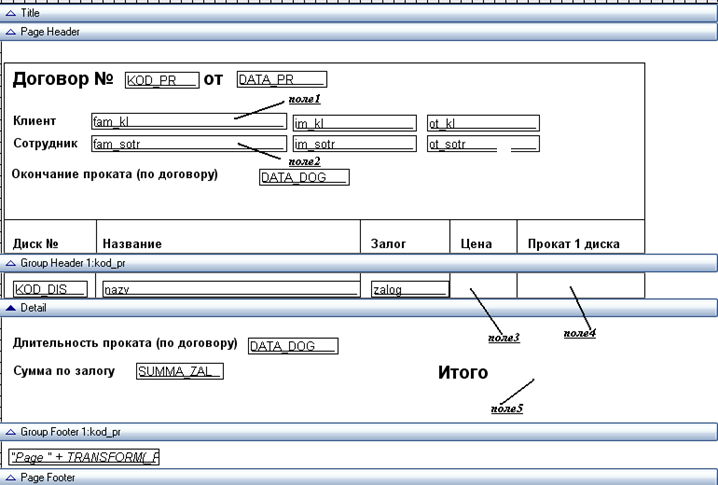


Рис. 116 – Макет отчета **dogovor**

1. Выполните просмотр отчета и убедитесь в правильности вычислений.
2. Запустите форму **oformlenie**, используйте кнопку **Print Report** и убедитесь в правильности создания появляющегося отчета.
3. Завершите работу с проектом.

# 6. СОЗДАНИЕ ЗАПРОСОВ И ЛОКАЛЬНЫХ ПРЕДСТАВЛЕНИЙ

## 6.1. Общие сведения о запросах и представлениях. Создание запросов и представлений

**Запрос** – это обращение к данным для получения выборочной информации и выполнения действий с данными. Запрос – динамический набор данных, которые существуют только во время выполнения запроса. Т. е. при каждом новом выполнении запроса можно получать различные результаты, учитывающие все изменения, которые произошли с данными в таблицах.

Запрос можно создать после создания и заполнения таблиц базы данных, а так же после создания схемы данных и организации связей между таблицами.

**Функции запроса:**

1. Выборка данных, соответствующих определенному критерию.
2. Просмотр и анализ данных.
3. Выполнение вычислений.
4. Подсчет итогов.
5. Объединение данных из разных таблиц.
6. Включение данных в форму или отчет.

Запросы отображаются в проекте на вкладке **Data** в группе **Queries**. По умолчанию результат запроса выглядит как таблица в режиме **Browse**. В таблице-результате нельзя менять данные.

**Способы создания запросов:**

1. Конструктор.
2. Мастер.

Наиболее удобным вариантом является использование конструктора (**New – New Query**). Этапы работы конструктора будут описаны при выполнении задания.

Каждому запросу соответствует программный код на языке SQL[[12]](#footnote-13). Этот код можно использовать при написании процедур и создании курсоров (временных таблиц).

**Панель инструментов QueryDesigner**

Панель инструментов (рис. 117) появляется при вызове конструктора. Если панель не отображается на экране, то ее можно добавить, используя команду меню **View** → **Toolbars** →**Query Designer**.



Рис. 117 – Панель инструментов **Query Designer**

– добавить к источнику данных новую таблицу



– удалить из источника данных выделенную таблицу



– добавить условие связи между таблицами



– получить код запроса на языке SQL



– скрыть или отобразить область настройки параметров запроса



 – открыть диалоговое окно для выбора, куда выводить результат запроса

**Представление** – это объект базы данных, в котором хранится информация о том, как должны быть выбраны (обработаны) данные и по каким критериям. По существу, представление – то же самое, что и запрос, но представление является объектом базы данных и может быть использовано при создании различных объектов проекта наравне с таблицами баз данных.

Представления отображаются в проекте на вкладке **Data** в группе **Local Views** в составе базы данных. По умолчанию результат запуска представления выглядит как таблица в режиме **Browse**. В результате-таблице данные могут быть изменены.

**Примечание** – Создание представлений и прочие операции с представлениями ведутся так же, как и с запросами.

На практике при формировании запросов и представлений часто используются выборки из нескольких таблиц, т. к. в реляционных базах данных информация содержится не в одной отдельной таблице, а в совокупности связанных таблиц.

При создании многотабличного запроса или представления в окно конструктора добавляются все участвующие в выборке таблицы и определяются условия их объединения.

Если между участвующими в запросе или представлении таблицами в базе данных установлены постоянные отношения, то в окне конструктора эта связь будет отображаться в виде линии, соединяющей таблицы, а на вкладке **Join** (Объединение) появится запись, содержащая условие объединения таблиц.

Если между таблицами, на основе которых строится запрос или представление, в базе данных отсутствует связь, то при добавлении таблиц к источнику данных на экране отобразится окно настройки связей **Join Condition** (рис. 118).

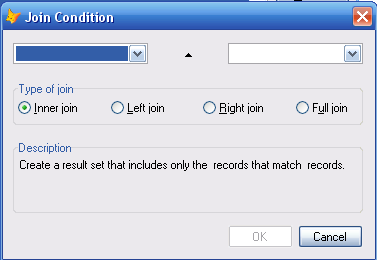


Рис. 118 – Окно настройки связей **JoinCondition**

В верхней части диалогового окна **Join Condition** размещены два раскрывающихся списка, содержащие поля, которые можно использовать для объединения таблиц[[13]](#footnote-14). Ниже расположен переключатель **Type of join** (Тип объединения), содержащий опции, определяющие тип создаваемой между таблицами связи. Назначение опций описано в табл. 16.

Таблица 16. Назначение опций диалогового окна **Join Condition**

|  |  |
| --- | --- |
| **Опция** | **Тип создаваемой связи[[14]](#footnote-15)** |
| **Inner join** (Внутреннее объединение)[[15]](#footnote-16) | Создает объединение, в котором выбираются только те записи, которые содержат совпадающие значения в полях связи |
| **Left join** (Объединение слева) | Создает объединение, в котором выбираются все записи из левой таблицы, а также записи из правой таблицы, значения поля связи которого совпадают со значениями поля связи левой таблицы |
| **Right join** (Объединение справа) | Создает объединение, в котором выбираются все записи из правой таблицы, а также записи из левой таблицы, значения поля связи которого совпадают со значениями поля связи правой таблицы |
| **Full join** (Полное объединение) | Создает объединение, в котором выбираются все записи из правой и левой таблиц |

### ЗАДАНИЕ 27. Создание запросов на основе одной таблицы

**Точное совпадение значений полей**

В результате выполнения задания будут созданы запросы:

**Disk\_vyd** – список дисков, находящихся в прокате

**Disk\_god** – список дисков, выпущенных в 2013 или 2014 году

1. В папке с проектом **DVD** создайте папку **Query**, которую будут сохраняться созданные запросы.
2. Запустите проект **dvd.pjx**.
3. Разработайте запрос, который выводит информацию о дисках, выданных в прокат. Для этого в проекте на вкладке **Data** выделите объект **Queries** и нажмите кнопку **New**. В появляющемся окне выберите вариант **New Query**, в результате чего будет запущен конструктор запросов**Query Disigner**. Первоначально поверх окна конструктора располагается окно выбора источника данных для запроса. Выберите в нем таблицу **Disk**.
4. Работа с конструктором запросов выполняется в несколько этапов с помощью вкладок нижней части окна:
   1. вкладка **Fields** (рис. 119) – выбор полей таблиц базы данных, которые будут отображаться при запуске запроса; добавьте поля, представленные на рис. 119, используя двойной щелчок левой кнопки мыши или кнопку **Add**;

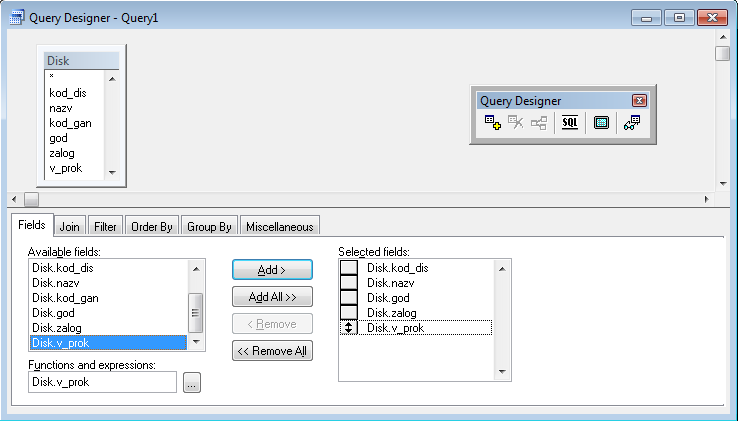


Рис. 119 – Вкладка **Fields** конструктора запросов

* 1. вкладка **Join** – используется для коррекции связи между таблицами, на основе которых создается запрос; поскольку в данном задании запрос будет создан на основе одной таблицы, то эта вкладка остается неиспользованной и пустой;
  2. вкладка **Filter** (рис. 120) – используется для выборки данных с помощью запроса; здесь следует выбрать поля и наложить на них условия отбора; установите условия отбора, как представлено на рис. 120: данные параметры обозначают «*поле* ***v\_prok*** *содержит значение* ***истина***»; чтобы изменить условие на «*поле* ***v\_prok*** *содержит значение* ***истина***», нажмите на кнопку в столбце **Not[[16]](#footnote-17)**(обратите внимание на изменение вида кнопки);

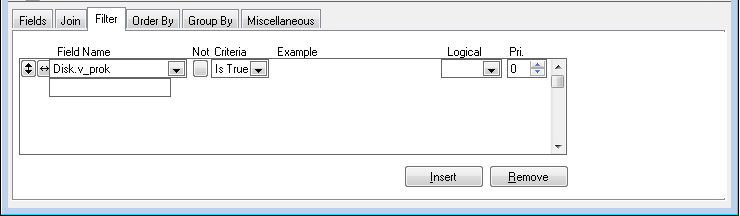


Рис. 120 – Вкладка **Filter** конструктора запросов

* 1. вкладка **Order By** (рис. 121) – используется для настройки сортировки отображаемой информации; установите параметры сортировки, как на рис. 121, для сортировки дисков по названию;

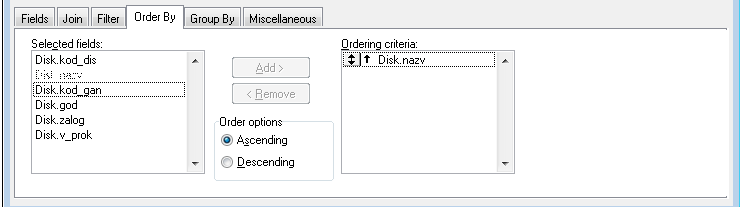


Рис. 121 – Вкладка **Order B**y конструктора запросов

* 1. вкладка **Group By** – используется для группировки данных, выводящихся в результате запроса; поскольку в данном задании запрос не требует группировки, то эта вкладка останется пустой;
  2. вкладка **Miscellaneouse**– позволяет задать дополнительные условия, такие как признак выборки повторяющихся значений, количество или процент выбора данных; поскольку в данном задании дополнительные параметры не используются, то эта вкладка останется пустой.

1. Теперь запрос создан. Выполните его: вызовите контекстное меню свободной области конструктора и выберите в нем команду **Run Query**. Убедитесь, что на экране результат отображается в виде, представленном на рис. 122[[17]](#footnote-18).

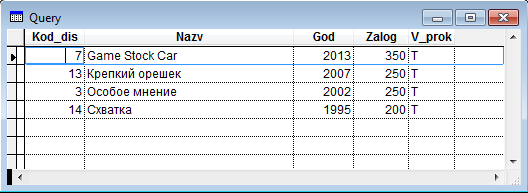


Рис. 122 – Результат запроса **Disk\_vyd**

Обратите внимание – это и есть стандартный результат запроса в режиме **Browse**. Здесь представлена информация о дисках, которые были выданыв прокат (v\_prok=F).

1. Закройте таблицу-результат.
2. При необходимости отобразите панель инструментов **Query Designer** (View → Toolbars →Query Designer), и нажмите на ней кнопку QueryDestination, в результате чего откроется окно выбора, куда выводить результаты запроса. На данный момент действует режим **Browse**. Нажмите кнопку Table (рис. 123). С помощью кнопки  выберите в качестве места сохранения папку **Query** и введите имя таблицы **disk\_vydl**. Нажмите кнопку ОК.

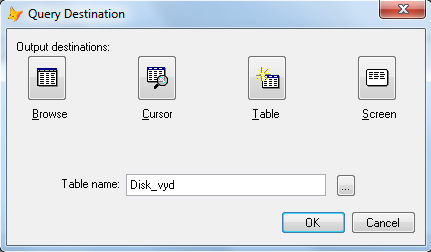


Рис. 123–Окно **Query Destination** для варианта **Table**

1. Снова выполните команду **Run Query** контекстного меню. Обратите внимание, что никакого результата на экране нет. Это произошло из-за последней настройки – результат запроса сохранился в таблицу.
2. Откройте папку **Query** и убедитесь, что в ней появилась таблица с именем **disk\_vyd.dbf[[18]](#footnote-19)**.
3. Снова нажмите кнопку  и измените вариант вывода результатов запроса на **Screen**. Обратите внимание, что результат отображается непосредственно в окне FoxPro.
4. Снова нажмите кнопку  и измените вариант вывода результатов запроса на **Browse**. Закройте конструктор запроса и в появляющемся окне запроса на сохранение нажмите кнопку **Yes**, выберите папку для сохранения **Query** и введите имя запроса **Disk\_vyd**. Обратите внимание, что в проекте в узле **Queries** появился созданный запрос.
5. Действуя аналогично, создайте запрос **Disk\_god**, отображающий информацию о дисках выпущенных в 2013 или 2014 году, данные должны быть отсортированы в порядке возрастания года выпуска:
   1. поля для отображения: код диска, название, год;
   2. условия отбора представлены на рис. 124:

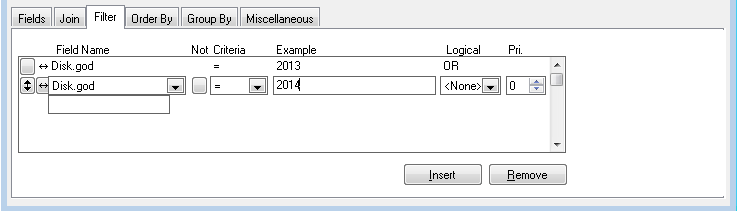


Рис. 124 – Условия отбора для запроса **Disk\_god**

* 1. содержимое вкладки **Order By** представлено на рис. 125:

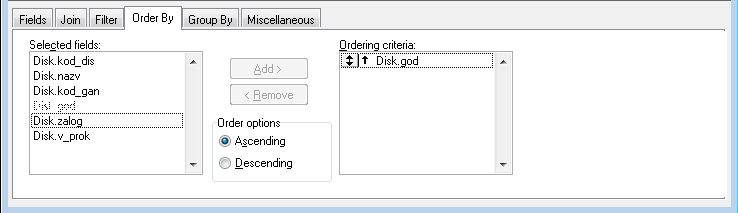


Рис. 125 – Условия сортировки для запроса **Disk\_god**

1. Запустите созданный запрос и убедитесь в правильности его создания. Обратите внимание на заголовки столбцов – они совпадают с именами соответствующих полей таблицы базы данных. Заголовки столбцов можно поменять, используя SQL-код запроса.
2. Откройте запрос **Disk\_god** в конструкторе (кнопка **Modify**).
3. Вызовите на свободной части запроса контекстное меню и выберите команду **View SQL** (или нажмите кнопку  панели инструментов). В результате выполнения команды появится окно, представленное на рис. 126[[19]](#footnote-20).

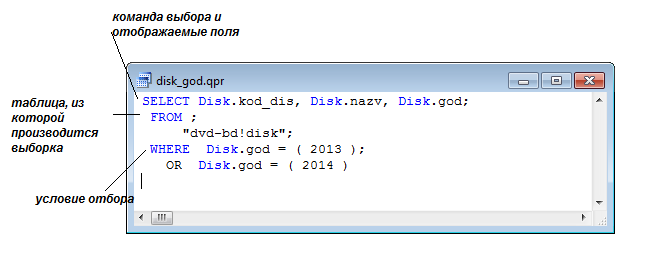


Рис. 126 – Окно SQL-кода запроса **Disk\_god**

1. Чтобы изменить подписи столбцов, отредактируйте код следующим образом (рис. 127)[[20]](#footnote-21):

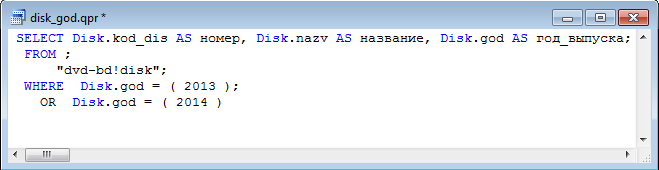


Рис. 127 – Отредактированный SQL-код запроса Disk\_god

1. Завершите работу с проектом.

**Значения полей лежат в пределах указанного диапазона**

В результате выполнения задания будет создан запрос **Disk\_zalog** – список дисков, залог за прокат которых находится в пределах от 280 до 350 рублей.

1. Запустите проект **dvd.pjx**.
2. Создайте запрос **Disk\_zalog**, отображающий информацию о дисках, залог по которым располагается в пределах от 280 до 350 рублей, данные должны быть отсортированы в порядке убывания залога:
   1. поля для отображения: код диска, название, залог;
   2. условия отбора представлены на рис. 128:

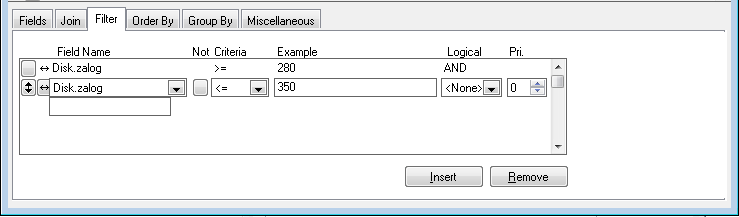


Рис. 128 – Условия отбора для запроса **Disk\_zalog**

* 1. содержимое вкладки **Order By** представлено на рис. 129:

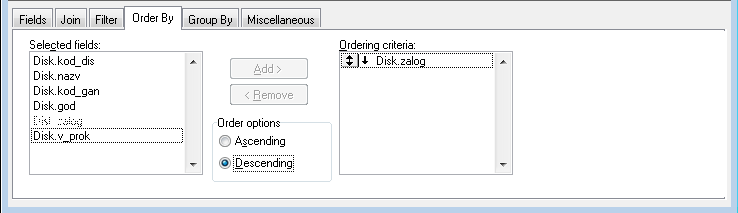


Рис. 129 – Условия сортировки для запроса **Disk\_zalog**

1. Запустите созданный запрос и убедитесь в правильности его создания.
2. Завершите работу с проектом.

**Выбор полей, не удовлетворяющих заданному условию**

В результате выполнения задания будет создан запрос **Disk\_nal** – список дисков в наличии.

1. Запустите проект **dvd.pjx**.
2. Создайте запрос **Disk\_nal**, отображающий сведения о дисках в наличии:
   1. поля для отображения: код диска, название, в прокате, год выпуска;
   2. условие отбора представлено на рис. 130 (обратите внимание на кнопку столбца **Not**– это и есть способ отобразить значения, не удовлетворяющие условию **v\_prok=.T.**, т. е. заданное условие позволяет отобразить информацию о дисках, не находящихся в прокате):

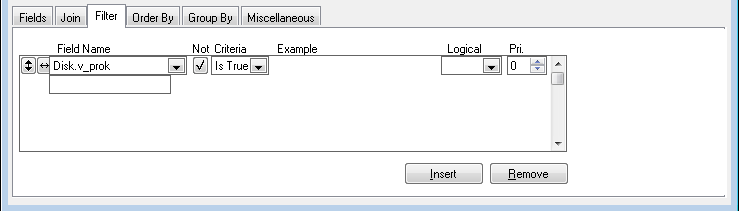


Рис. 130 – Условие отбора для запроса **Disk\_nal**

1. Сохраните созданный запрос под именем **Disk\_nal** и проверьте правильность его выполнения.
2. Завершите работу с проектом.

### ЗАДАНИЕ 28. Создание локального представления на основе нескольких таблиц

В результате выполнения задания будет создано представление, содержащее информацию и дисках, соответствующих жанрах и категориях дисков. Такой набор информации не хранится ни в одной таблице баз данных полностью, поэтому локальное представление будет создано на основе трех таблиц: **Disk**, **Ganr**, **Kategor**. В дальнейшем это представление будет использовано для создания прайс-листа дисков DVD-проката.

1. Запустите проект **dvd.pjx**.
2. В менеджере проекта выделите объект **LocalViews** и нажмите кнопку **New**. В появляющемся окне запроса способа создания выберите вариант **NewView**. В результате будет запущен конструктор представлений, работа с которым аналогична работе с конструктором запросов. Первоначально на экране поверх окна конструктора появится запрос на добавления таблиц к источнику данных запроса (рис. 131). И здесь при добавлении таблиц возникнут некоторые нюансы.

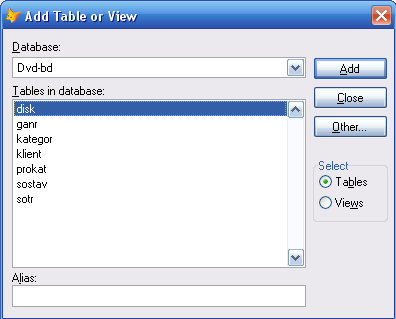


Рис. 131 – Окно добавления таблиц к источнику данных[[21]](#footnote-22)

* 1. выделите таблицу **Disk**, нажмите кнопку **Add**;
  2. выделите таблицу **Kategor**, нажмите кнопку **Add**; обратите внимание, что на экране появляется окно настройки связей **JoinCondition** (рис. 132). Оно появилось из-за того, что в базе данных между таблицами **Kategor** и **Disk** нет связи (поэтому здесь следует нажать кнопку **Cancel**)[[22]](#footnote-23);

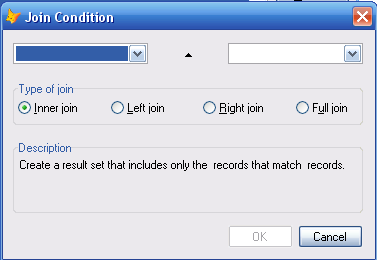


Рис. 132 – Окно настройки связей **JoinCondition**

* 1. выделите таблицу **Ganr** и нажмите кнопку **Add**; убедитесь, что источник данных представления принял вид, представленный на рис. 133 (при необходимости добавьте связи):

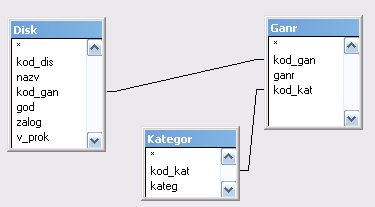


Рис. 133 – Источник данных представления

* 1. настройте содержимое вкладки **Fields** конструктора представлений (рис. 134)[[23]](#footnote-24):

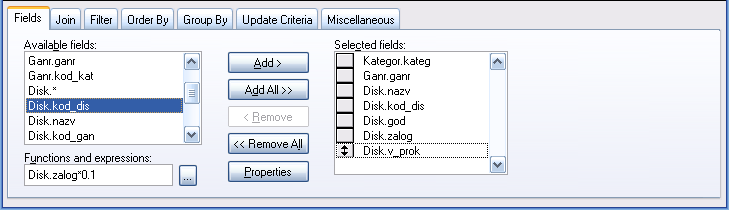


Рис. 134 – Содержимое вкладки **Fields**

* 1. настройте сортировку данных так, чтобы категории выводились по алфавиту, внутри каждой категории жанры тоже выводились в алфавитном порядке, а в каждом жанре по алфавиту отображались названия дисков (рис. 135) (*обратите внимание на порядок отображения полей сортировки*):

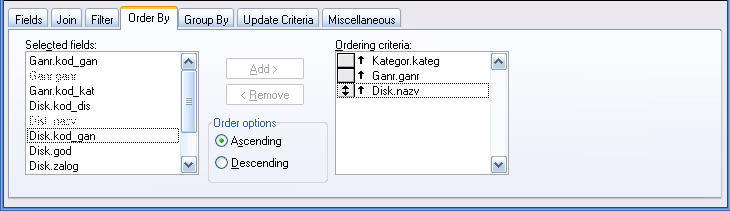


Рис. 135 – Настройка полей сортировки

* 1. выполните настройку обновления представления при изменении записей в таблицах базы данных, используя флажок **SendSQLupdates**вкладки **UpdateCriteria** (рис. 136);

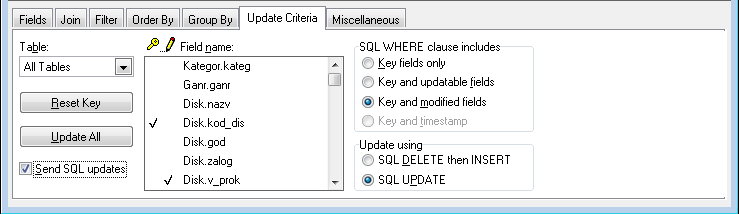


Рис. 136 – Настройка параметров обновления

* 1. выполните просмотр результатов представления и сохраните представление под именем **Diski\_vse**.

1. Просмотрите SQL-код представления, обратите внимание на его основную часть (рис. 137)[[24]](#footnote-25).

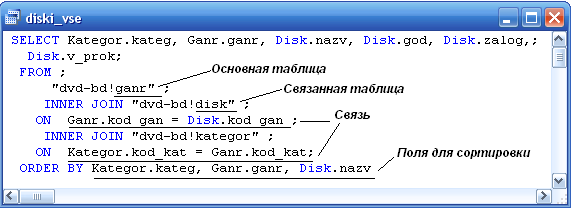


Рис. 137 – SQL-код представления **Diski\_vse**

1. Убедитесь, что в менеджере проекта в узле **LocalViews** появилось созданное представление.
2. Завершите работу с проектом.

## 6.2. Вычисления в запросах и представлениях

Visual FoxPro позволяет не только включать в результирующую таблицу запроса информацию из исходных таблиц, но и производить вычисления над ними, предоставляя, таким образом, возможность получить данные, отсутствующие в исходной таблице.

Вычисляемые поля в запросе могут быть двух видов:

* Для каждой строки запроса.
* Итоговое вычисление (одно результирующее поле на весь запрос)

Например, в таблице **Prokat** есть дата выдачи дисков и договорная дата возврата дисков. На их основе в запросе или представлении можно получить поле длительности проката как разность указанных полей. Оно будет подсчитано для *каждой* строки запроса (для каждого договора).

Кроме того, вы можете использовать вычисляемые поля для объединения нескольких полей исходной таблицы в одно результирующее поле. Например, в трех разных таблицах хранится информация о диске, соответствующем жанре и категории. В результирующий запрос или представление вы можете поместить одно поле, в котором будет размещены и название, и жанр, и категория. Оно будет рассчитываться для каждой строки запроса (для каждого диска).

Если вы хотите подсчитать общую сумму по всем договорам проката, найденным в запросе, то это *итоговое* вычисление. Оно – одно на весь запрос.

Для формирования выражения предназначено поле **Functionsandexpressions** (Функции и выражения) вкладки **Fields** (Поля). Нажмите расположенную справа от поля кнопку вызова построителя выражения и в диалоговом окне **ExpressionBuilder** (Построитель выражения) создайте выражение для вычисляемого поля. После того как выражение в поле **Functionsandexpressions** (Функции и выражения) сформировано, нажатием кнопки **Add** (Добавить) перенесите его в список **Selectedfields** (Выбранные поля).

Замечание: Если в запросе (представлении) предполагается наличие поля с итоговым вычислением, например **sum(prokat.summa\_zal+0.1\*prokat.summa\_zal\*(prokat.data\_dog-prokat.data\_pr))**, то в группе **SelectedFields** следует расположить только одно это поле.

### ЗАДАНИЕ 29. Создание вычисляемых полей в представлениях (запросах)

В результате выполнения задания будет отредактировано представление **Diski\_vse**, в котором не отображалась цена проката диска за сутки, зависящая от залога и вычисляемая по формуле (**1**). Вычисляемое поле можно создать, используя вкладку **Fields** конструктора представлений или отредактировав SQL-код.

1. Запустите проект **dvd.pjx**.
2. Откройте для редактирования в конструкторе представление **Diski\_vse**.
3. Добавьте поле для вычисления цены проката за сутки:
   1. на вкладке **Fields** установите курсор в поле **Functionsandexpressions** и нажмите кнопку  для вызова построителя выражений;
   2. в построителе измените таблицу на **Disk** (рис. 138), т.к. вычисляемое поле зависит от поля **Disk.zalog**;

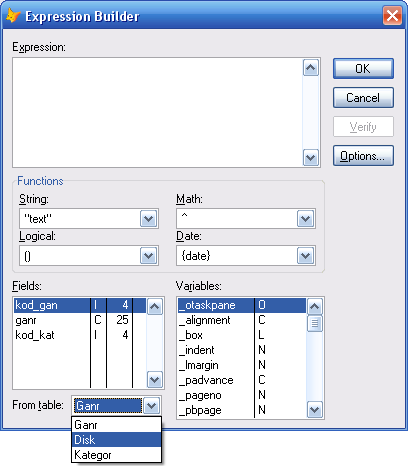


Рис. 138 – Построитель выражений: изменение таблицы

* 1. в окне построителя выполните двойной щелчок по полю **zalog** в списке **Fields**, в результате чего оно отобразится в области **Expression**; отредактируйте выражение в соответствии с формулой (1) (рис. 139) и нажмите кнопку **ОК**;

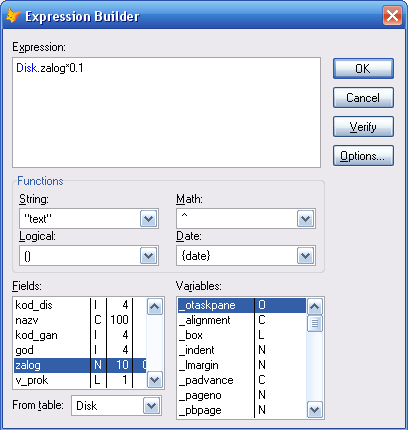


Рис. 139 – Выражение для вычисляемого поля в построителе

* 1. теперь введенное выражение отображается в конструкторе в поле **Functionsandexpressions**; убедившись, что курсор так и находится в этом поле, нажмите на кнопку **Add** – вычисляемое поле добавится в представление.

1. Запустите представление и убедитесь в появлении вычисляемого столбца.
2. Завершите работу с проектом.

### ЗАДАНИЕ 30. Выполнение итоговых вычислений в представлении или запросе

В результате выполнения задания будут созданы запросы, которые подсчитывают:

* общую сумму по всем договорам проката, оформленным, например 26.10.2014;
* общее количество оформленных в этот день договоров;
* и общее количество, и общую сумму по договорам, оформленным 26.10.2014.

При подсчете количества договоров будет использоваться уже знакомая по работе с отчетами итоговая функция **COUNT()**; поле, по которому будет вестись подсчет, – поле первичного ключа таблицы:

**Prokat.kod\_pr**[[25]](#footnote-26).

При подсчете общей суммы, на которую оформлены договора, будет использоваться итоговая функция **SUM()**, также знакомая по работе с отчетами; поле, по которому будет подводиться итог, вычисляемое, позволяющее найти сумму по каждому прокату:

**Prokat.summa\_zal+0.1\*Prokat.Summa\_zal\*(Prokat.data\_dog-Prokat.data\_pr)**.

1. Запустите проект **dvd.pjx**.
2. Запустите форму **oformlenie.scx** и обратите внимание на договора, оформленные 26.10.2014. Запомните их количество.
3. В окне **Command** введите команду **SETDATEGERMAN**, чтобы изменить формат даты.
4. Вызовите конструктор запросов.
5. Создайте запрос, отображающий информацию о количестве договоров проката, оформленных в указанный день:
   1. в область данных запроса добавьте таблицу **Prokat** (ЗАДАНИЕ 27, п.3);
   2. на вкладке **Fields**в области **AvailableFields** выделите поле **Prokat.kod\_pr**;
   3. установитекурсорвполе**Functions and expressions**;
   4. нажмите на кнопку , в результате чего откроется построитель выражений **ExpressionBuilder**;
   5. скопируйте содержимое поля в буфер обмена (**Ctrl+C**);
   6. в списке **Math** выберите функцию **Count()** и вставьте ее в поле **Expression**, выполнив на ней щелчок левой кнопкой мыши;
   7. отредактируйте выражение, вставив содержимое буфера обмена (**Ctrl+V)**,чтобы оно приняло вид **Count(Prokat.kod\_pr)**;
   8. закройте построитель, убедитесь, что курсор находится в поле **Functionsandexpressions**и нажмите кнопку **Add** – убедитесь, что вычисляемое поле появилось в списке **SelectedFields**; теперь в запросе появится поле с итоговым количеством;
   9. перейдите на вкладку **Filter**: в списке **FieldName** выберите **Prokat.data\_pr**, в списке **Criteria** – операцию **=**, в поле **Example**введите выражение **ctod(“26.10.2014”)**.[[26]](#footnote-27)
6. Запустите запрос и убедитесь в правильности его выполнения – результат содержит всего одну строку и одно поле, отображающее информацию о количестве договоров прокатов, оформленных 26.10.2014.
7. Сохраните запрос под именем **Kol\_data**.
8. Снова запустите конструктор запроса.
9. Создайте запрос, определяющий общую сумму по договорам проката, оформленным 26.10.2014: действия аналогичны п. 5, но вычисляемое поле, которое следует добавить на вкладке Fields, – **Sum(Prokat.summa\_zal+0.1\*Prokat.Summa\_zal\*(Prokat.data\_dog-Prokat.data\_pr))**.
10. Запустите запрос – результат содержит всего одну строку и одно поле, отображающее информацию об общей сумме по договорам проката, оформленным 26.10.2014.
11. Просмотрите SQL-код запроса – его в дальнейшем можно использовать в качестве процедуры, наложенный на событие объекта формы.
12. Сохраните запрос под именем **Sum\_data**.
13. Создайте еще один запрос, который определит и общее количество, и общую сумму по договора проката, оформленным 26.10.2014. Его создание аналогично созданию двух предыдущих договоров, но на вкладке **Fields** нужно добавить два вычисляемых поля (п.5, п.9).
14. Проверьте правильность работы запроса и сохраните его под именем **Kol\_Sum**.
15. Завершите работу с проектом.

### ЗАДАНИЕ 31. Использование представления для создания отчета

При выполнении ЗАДАНИЯ 22 был создан отчет о дисках с группировкой по жанрам, но группировка по категориям в нем была невозможна. При выполнении ЗАДАНИЙ 28 и 29 было разработано локальное представление**diski\_vse**, содержащее информацию обо всех дисках DVD-проката. Теперь на его основе можно разработать печатный документ **Каталог дисков**, в котором информация о дисках будет сгруппирована по жанрам и категориям.

1. Запустите проект **dvd.pjx**.
2. Запустите мастер создания отчетов для создания отчета на основе одной таблицы.
3. Обратите внимание, что на первом шаге мастера в списке таблиц стало доступно созданное ранее представление **Diski\_vse** (рис. 140):

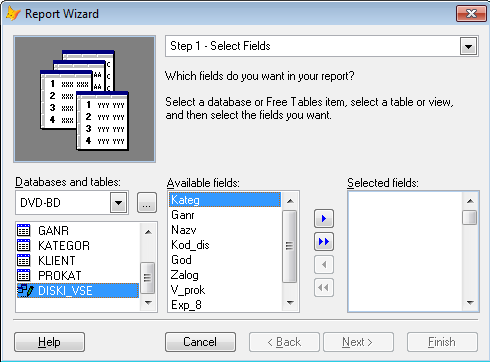


Рис. 140 – Появление представления в списке доступных таблиц для отчета

1. Выберите в отчет все поля представления.
2. Установите параметры группировки, чтобы все жанры были сгруппированы по категориям, а внутри каждого жанра располагалась соответствующая группа дисков (рис. 141).

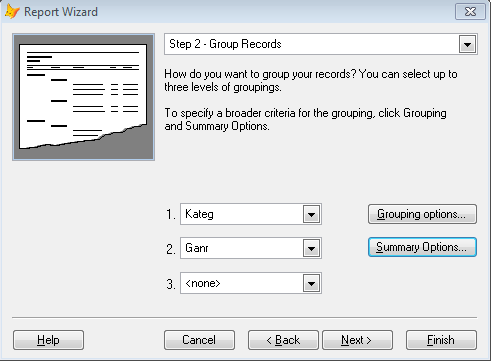


Рис. 141 – Параметры группировки для каталога дисков

1. Установите параметры итоговых вычислений (рис. 142):

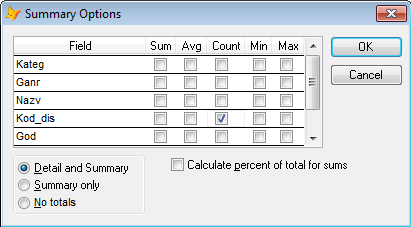


Рис. 142 – Установка параметров итоговых вычислений

1. Выберите стиль отчета **Ledger**.
2. Параметры сортировки можно не указывать, т.к. сортировка записей была указана при создании соответствующего представления.
3. Сохраните отчет в папке **Reports** под именем **Katalog.frx**.
4. Выполните просмотр отчета и убедитесь в правильности его создания.
5. Подобно действиям ЗАДАНИЙ 23 и 26 приведите отчет к виду, представленному на рис. 143; макет отчета представлен на рис. 144[[27]](#footnote-28).

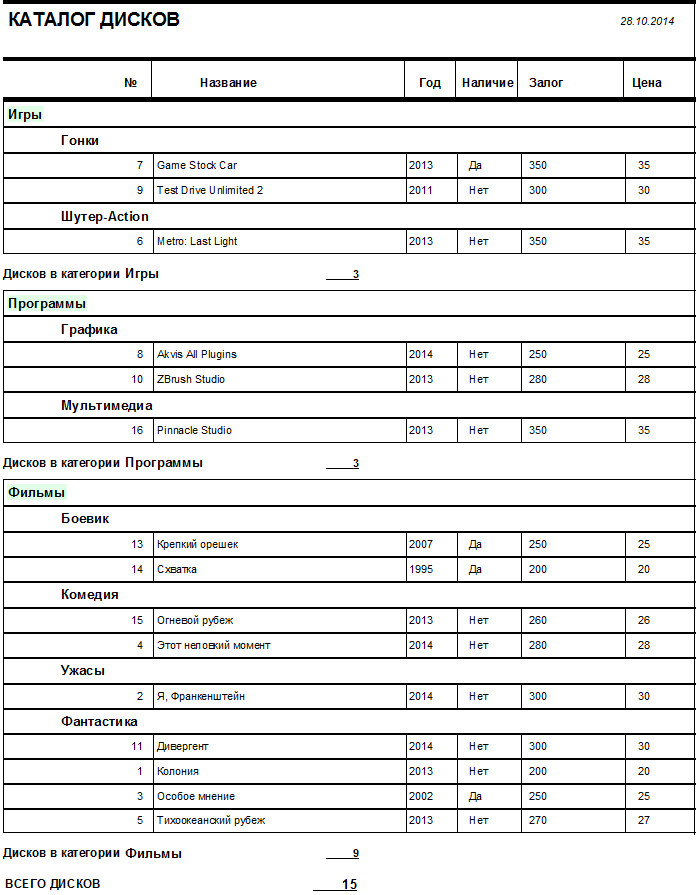


Рис. 143 – Вид отчета **Katalog**

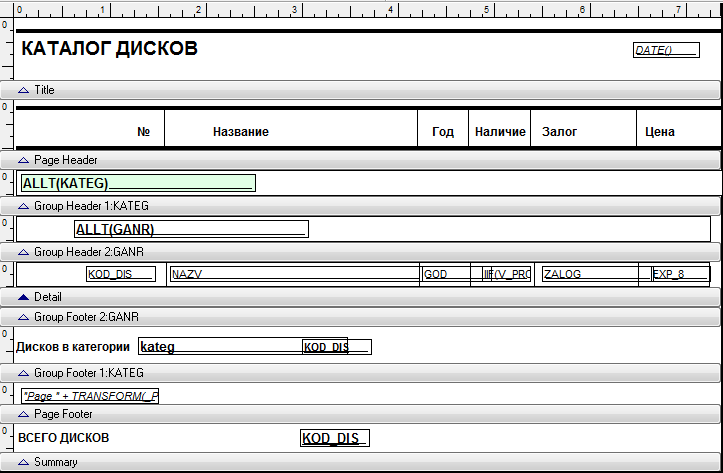


Рис. 144 – Макет отчета **Katalog**

1. Проверьте правильность разработки отчета. Закройте отчет.
2. Данный отчет можно использовать для отображения информации о дисках, выданных на руки, используя фильтрацию данных (подобно ЗАДАНИЮ 25). В окне **Command**обратите внимание на последнюю команду, отредактируйте ее и запустите на выполнение:

REPORT FORM d:\dvd\reports\katalog.frx FORv\_prok PREVIEW

1. Подобным образом отчет можно использовать и для вывода информации о дисках в наличии. Отредактируйте команду и запустите ее на выполнение:

REPORT FORM d:\dvd\reports\katalog.frx FOR NOT v\_prok PREVIEW

1. Завершите работу с проектом.

# 7. РАЗРАБОТКА СРЕДСТВ ОБРАБОТКИ ДАННЫХ НА ОСНОВЕ ОПИСАНИЯ ПРЕДМЕТНОЙ ОБЛАСТИ И ПОТРЕБНОСТЕЙ ПОЛЬЗОВАТЕЛЯ

В данном разделе будут разработаны объекты проекта, необходимость в которых обусловлена потребностями пользователя и предметной областью. Набор этих объектов очень разнообразен, их количество и назначение зависят о разработчика. Разработчик определяет, какие запросы будут реализованы в проекте, какие приемы обработки данных, в каком виде будут выданы результаты обработки данных.

Определимся с некоторым минимумом подобных объектов проекта.

Поскольку диски выдаются в прокат на определенный срок и должны быть возвращены, то разумным было бы определять клиентов, которые не вернули диски вовремя. Или которые должны их вернуть сегодня (чтобы с клиентами можно было связаться). Результатом подобного поиска может быть таблица в режиме Browse, форма (например, с таблицей Grid), отчет.

В проекте разработана форма для редактирования договора проката. При запуске формы отображаются данные обо всех договорах, начиная с первого. А если необходимо отредактировать договор с номером 1025? Конечно, можно воспользоваться стандартной кнопкой поиска на панели управляющих кнопок формы. Но оно англоязычное и не совсем понятное обычному пользователю, поэтому было бы неплохо как-то видоизменить форму, чтобы появлялся русскоязычный запрос номера договора для редактирования.

Достаточно часто в разработках встает необходимость получения итоговых сведений по временным интервалам: об оформленных договорах, о продажах, о поставках, о заказах и т.д. Поэтому можно разработать форму, запрашивающую интервал дат и позволяющую вывести итоговый отчет.

Чтобы наглядно убедиться в правильности разработки будущих запросов, отчетов и форм, необходимо внести в базу данных больше сведений. Достаточно наполнить записями таблицу **Prokat**.

Запустите форму **oformlenie.scx** и добавьте следующие данные о прокатах дисков (выбирайте любого клиента и сотрудника):

|  |  |  |  |
| --- | --- | --- | --- |
| **№ договора** | **Дата оформления** | **Дата возврата по договору** | **Выданные диски** |
| 4 | 24.10.2014 | сегодня[[28]](#footnote-29) | 5,10 |
| 5 | 24.10.2014 | 27.10.2014 | 1 |
| 6 | 26.10.2014 | 05.11.2014 | 6 |
| 7 | 26.10.2014 | 27.10.2014 | 7,8 |
| 8 | 27.10.2014 | 03.11.2014 | 9 |
| 9 | 27.10.2014 | сегодня[[29]](#footnote-30) | 11 |

Запустите форму **redakt.scx** и добавьте следующие данные о возврате дисков:

|  |  |
| --- | --- |
| **№ договора** | **Дата возврата фактическая** |
| 6 | 01.11.2014 |
| 8 | 05.11.2014 |

### ЗАДАНИЕ 32. Поиск информации о клиентах-должниках

Прежде всего, следует определиться, какую информацию необходимо получить в результате такого поиска и в каком виде она должна быть выдана. Во-первых, это информация о клиенте, в т.ч. его телефон, чтобы иметь возможность связаться с ним и предупредить о долге. Во-вторых, это информация о прокате: например, дата оформления и дата возврата дисков по договору. Для отображения таких данных можно создать форму или отчет, в которых в табличной части будут отображаться необходимые сведения (один договор соответствует одному клиенту). Возможно, по каждому должнику будет полезно иметь информацию о дисках, которые он взял. Тогда результат удобно оформить в виде формы или отчета с основной и подчиненной частью. В любом случае потребуется объединить данные таблиц **Prokat** и **Klient**, и выбрать из них записи, где поле **data\_fakt** не заполнено (ведь оно заполняется только при возврате дисков), а в поле **data\_dog** стоит дата, меньшая текущей (т.е. день возврата уже прошел). Для этого необходимо создать представление.

1. Запустите проект **dvd.pjx**.
2. Запустите конструктор представлений и в качестве источника данных укажите две таблицы: **Prokat** и **Klient**.
3. На вкладке **Fields** выберите поля[[30]](#footnote-31):

Prokat.kod\_pr

Prokat.kod\_kl

Prokat.data\_pr

Prokat.data\_dog

Prokat.data\_fakt

Prokat.summa\_zal

Klient.fam\_kl

Klient.im\_kl

Klient.ot\_kl

Klient.tel\_kl

Klient.g\_kl

Klient.ul\_kl

Klient.d\_kl

Klient.k\_kl

Klient.kv\_kl

Klient.ser

Klient.nom

1. Навкладке**Updates Criteria**установитефлажок**Send SQL updates**.
2. На вкладке**Filter** установите 2 критерия отбора:
   1. Фактическая дата возврата не заполнена, т.е. поле **data\_fakt** пустое. Для этого проверки можно воспользоваться функцией **Empty()**[[31]](#footnote-32): условие отбора **Empty(Prokat.data\_fakt) istrue**. Чтобы ввести в столбец **FieldName**выражение, раскройте соответствующий список (рис. 145) и выберите вариант **Expression**. В результате откроется окно построителя выражений (рис. 146), в котором введите выражение **Empty(prokat.data\_fakt**), используя функцию из списка **Logical**, или введите вручную. Установите в столбце **Criteria** значение **Is True**.

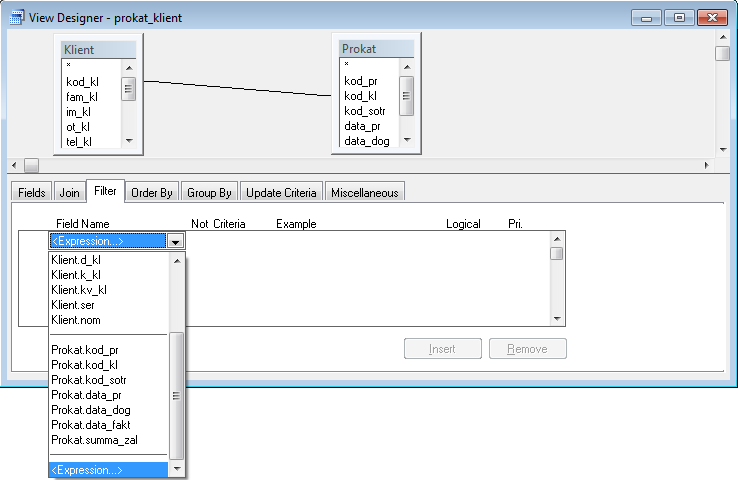


Рис. 145 – Создание критерия отбора на основе выражения

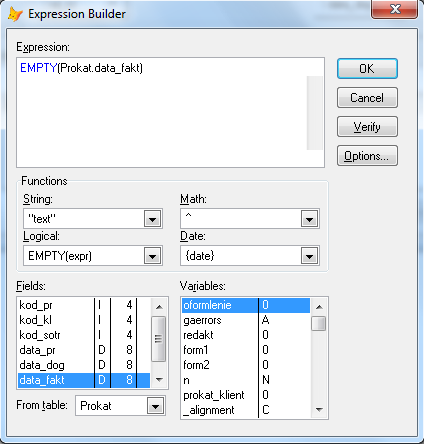


Рис. 146 – Построитель выражений в конструкторе представлений

* 1. Добавьте второе условие отбора: дата возврата по договору меньше текущей (рис. 147).

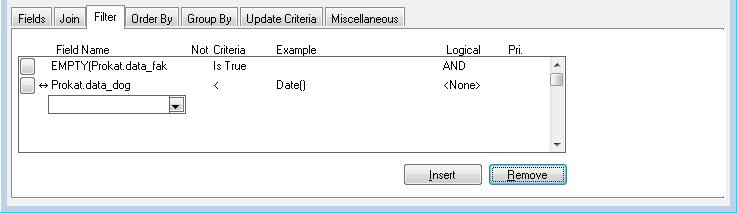


Рис. 147 – Условия отбора для представления **Prokat\_Klient**

1. Запустите представление, найдите столбцы **Data\_dog** и **Data\_fakt** и убедитесь в правильности выполнения задания. Обратите внимание на номера договоров проката – они должны появиться в создаваемой далее форме[[32]](#footnote-33).
2. Если даты в представлении отображаются в формате **мм/дд/гггг**, то измените их формат на привычный **дд.мм.гггг**, выполнив в окне **Command** команду **SetDateGerman**.
3. Сохраните представление под именем **Prokat\_Klient**.
4. Запустите форму **klient** или любую другую форму. Обратите внимание на появившееся сообщение (рис. 148). Оно говорит о том, что в рабочей области осталась открыта таблица, используемая в форме. Это последствие просмотра результатов запросов, представлений, таблиц в режиме **Browse** (поэтому результаты представлений лучше выводить в виде формы или отчета). Чтобы устранить данную ошибку введите в окно **Command**команду **Use** и нажмите клавишу **Enter**. Выполните попытку снова запустить форму. Если ошибка не устранена, то введите команду **CloseTables**. Снова повторите попытку просмотра формы. Если ошибка все еще возникает, то закройте проект и откройте его снова командой меню **File** – **Open**.

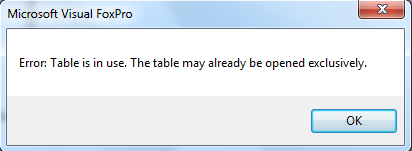


Рис. 148 – Сообщение об открытой в рабочей области таблице

1. Создайте первую форму, отображающую сведения о должниках (без информации о дисках). Запустите конструктор форм (**New** – **NewForm**). На панели инструментов **FormControls** выберите объект **Grid** и разместите его на форме протяжкой мыши (рис. 149).

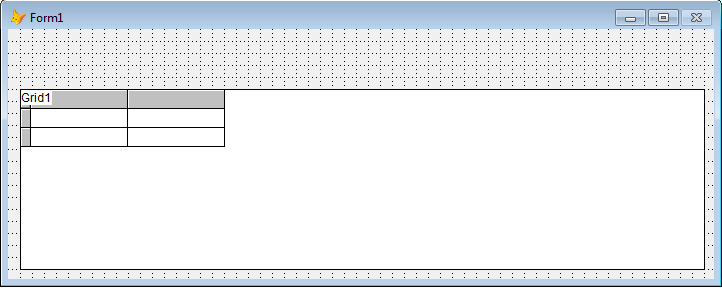


Рис. 149 – Размещение **Grid** на форме

1. Вызовите окно **DataEnvironment**, вызовите на нем контекстное меню и выполните команду **Add** для добавления источника данных (рис. 150). Обратите внимание, что в области **Select** выбран переключатель **Views**. Добавьте представление **Prokat\_Klient** в источник данных.

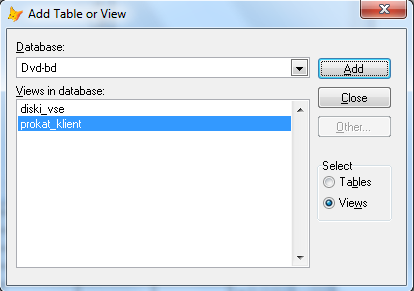


Рис. 150 – Выбор локального представления в качестве источника данных формы

1. Выделите **Grid** и в панели **Properties**на вкладке **All**измените свойство **ColumnCount** = **8**.
2. На вкладке **Data** измените свойство **RecordSource** = **Prokat\_Klient**.
3. Аналогично действиям ЗАДАНИЯ 11, выбирая поочередно столбцы созданной таблицы, установите для них свойство **ControlSource**:

**Column1** – **Prokat\_Klient.fam\_kl**

**Column2** – **Prokat\_Klient.im\_kl**

**Column3** – **Prokat\_Klient.ot\_kl**

**Column4** – **Prokat\_Klient.tel\_kl**

**Column5** – **Prokat\_Klient.kod\_pr**

**Column6** – **Prokat\_Klient.data\_pr**

**Column7** – **Prokat\_Klient.data\_dog**

**Column8** – **Prokat\_Klient.data\_fakt**

1. Аналогично действиям ЗАДАНИЯ 11, измените заголовки столбцов на**Фамилия**, **Имя**, **Отчество**, **Телефон**, **Договор**, **Дата выдачи**, **Дата возврата**, **Фактический возврат** (свойство **Caption**для объектов **Header1**).
2. На свободной части формы вызовите контекстное меню и выберите команду **RunForm**. В появляющемся запросе сохранения формы нажмите кнопку **Yes** и сохраните форму в папке **Forms** под именем **Dolgnik.scx**.
3. Убедитесь в правильности работы формы.
4. При необходимости отредактируйте в конструкторе ширину столбцов **Grid**, чтобы полностью были видны содержащиеся в ячейках данные.
5. Скройте последний столбец с помощью свойства **Visible** (столбец был необходим только для проверки правильности работы формы).
6. Приведите форму к виду, представленному на рис. 151:
   1. изменено свойство **Caption**формы;
   2. добавлена надпись **Список должников на**;
   3. добавлено поле, в котором отображается текущая дата (добавлено текстовое поле и в событие **refresh** формы введена команда **thisform.text1.value=date()**);
   4. для отображения дат в формате дд.мм.гггг в событие **Load**формы добавлена команда **SetDateGerman**.



Рис. 151 – Вид формы **Dolgnik**

1. Создайте вторую форму для отображения информации о должниках, но в ней будут отображаться сведения о выданных дисках по каждому договору. Для этого запустите мастер форм, выбрав вариант создания **One-to-ManyFormWizard**для создания формы на базе представления **Prokat\_Klient** и таблицы **Sostav.**
2. На первом шаге мастера выберите поля главной таблицы – представления **Prokat\_Klient**(рис. 152).

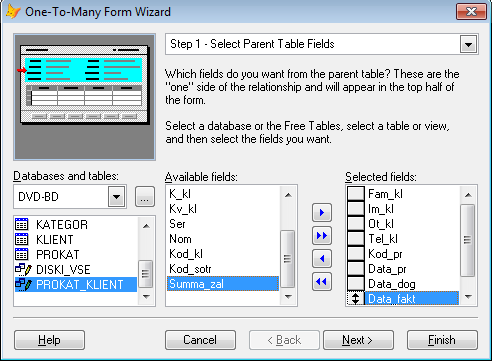


Рис. 152 – Выбор полей представления

1. На втором шаге укажите поля подчиненной таблицы (рис. 153).

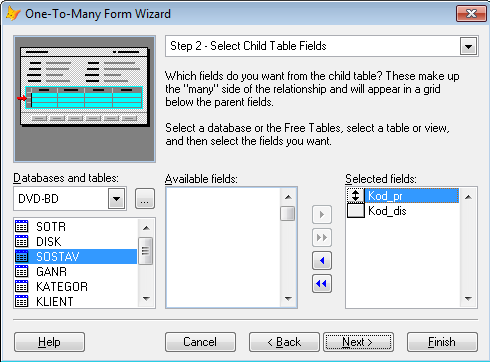


Рис. 153 – Выбор полей подчиненной таблицы

1. На остальных шагах выполняйте те же действия, что и при использовании мастера в ЗАДАНИИ 10. Сохраните форму в папке **Forms**под именем **Dolgnik-1.scx**.
2. Запустите созданную форму и убедитесь в правильности ее работы, контролируя содержимое верхней части формы (вне Grid).
3. Повторяя действия ЗАДАНИЯ 15 (п.1 – п.3), отредактируйте Grid так, чтобы столбцы **kod\_dis** и **kod\_pr** были скрыты (не удалены!), и в Grid были добавлены столбцы для отображения информации о дисках, как представлено на рис. 154.
4. Отредактируйте внешний вид формы в соответствии с рис. 154:
   1. добавлена надпись и вычисляемое текстовое поле с текущей датой, как в п.20;
   2. выполнена настройка отображения даты в формате дд.мм.гггг (как в п.20);
   3. на панели кнопок скрыты кнопки печати, поиска, добавления, изменения и удаления записей: панель переведена в режим редактирования командой контекстного меню **Edit** (кнопки доступны для выбора по отдельности), и для перечисленных кнопок изменено свойство **visible**.

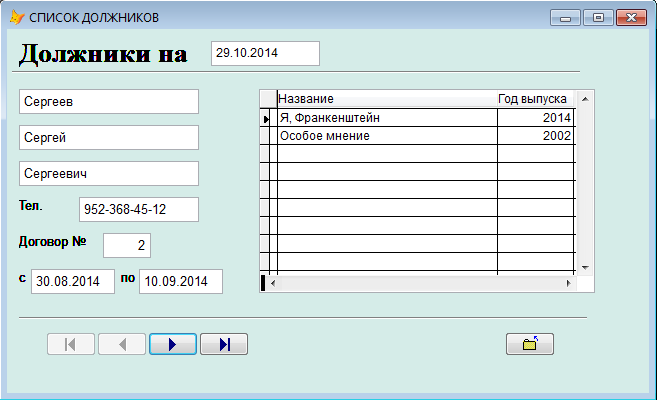


Рис. 154 – Внешний вид формы **Dolgnik-1**

1. Завершите работу с проектом.

### ЗАДАНИЕ 33. Применение фильтрации к форме

В результате выполнения задания будут внесены изменения в процедуру вызова формы редактирования договора проката (**redakt.scx**). В описании задачи говорится, что клиент будет возвращать диски и предъявлять при этом договор проката. Работник проката отыщет данные именно по этому договору и внесет в него изменения, отметив дату возврата дисков. Одним из вариантов реализации такой ситуации может послужить разработанное диалоговое окно, в котором запрашивается номер договора для редактирования. После ввода номера открывается форма **redakt**, содержащая информацию именно об этом договоре.

1. Запустите проект **dvd.pjx**.
2. Создайте форму, которая и будет выдавать запрос номера договора для редактирования:
   1. запустите конструктор форм;
   2. разместите на форме надпись (**Label**),поле (**TextBox**) и кнопку (**CommandButton**) (рис. 155);

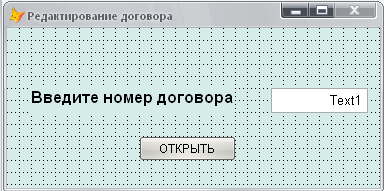


Рис. 155 – Форма запроса номера договора

* 1. измените тип данных поля **Text1** на **Numeric** (ЗАДАНИЕ 18, п.12);
  2. для события **Click** кнопки создайте процедуру (рис. 156); здесь в переменную N сохраняется номер договора, введенный пользователем, и запускается соответствующая форма.

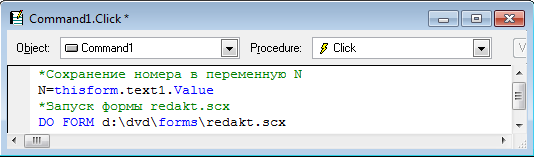


Рис. 156 – Процедура для события **Click** кнопки

1. Сохраните форму под именем **zapros.scx** в папке **Forms**.
2. Наложите фильтр на форму **redakt.scx**. Для этого откройте ее в конструкторе и в событие **Load** формы добавьте строку (рис. 157); здесь накладывается фильтр – отображать в форме записи базы данных, для которых ключевое поле **kod\_pr** содержит значение **N** (**kod\_pr** – первичный ключ таблицы, которая является главной для формы).

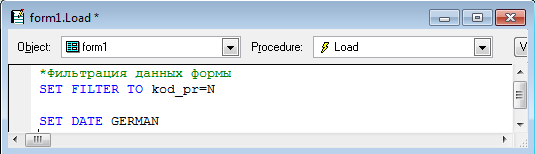


Рис. 157 – Процедура для события **Load** формы

1. Сохраните изменения в форме.
2. Запустите форму **zapros.scx**, введите номер договора **3** и нажмите кнопку**Открыть**. Обратите внимание, что в форме **redakt** отображаются данные именно этого договора. Закройте форму **redakt**.
3. Очистите поле ввода номера договора (там будет отображаться значение 0) и снова нажмите кнопку**Открыть**. Обратите внимание, что открылась пустая форма **redakt** – вообще говоря, это неверно. Логичнее в таком случае сообщить пользователю о том, что поле ввода номера договора обязательно должно быть заполнено. Для этого отредактируйте процедуру для события **Click** кнопки**Открыть** (рис. 158).

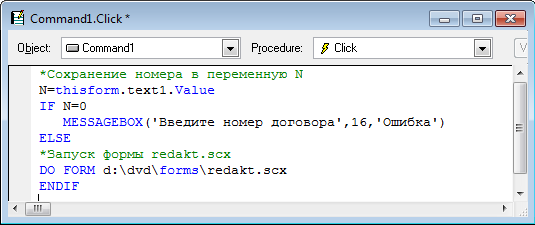


Рис. 158 – Отредактированная процедура для события **Click** кнопки

1. Сохраните изменения в форме и запустите ее.
2. Нажмите кнопку**Открыть**, не вводя номер договора. Убедитесь в правильности работы формы.
3. Завершите работу с проектом.

### ЗАДАНИЕ 34. Создание отчета о договорах проката, оформленных за период

В результате выполнения задания будут разработаны форма и отчет, позволяющие вывести сведения о договорах проката, оформленных за временной период или за указанную дату, причем будет подсчитана общая сумма по найденным договорам, а так же их количество. Следует отметить, что разработка формы достаточно сложна, т.к. на форме появится ранее не используемый объект**PageFrame**, в форме будут использованы программные процедуры, а также организованы вычисления с помощью SQL-запросов. Сам результат выборки данных будет оформлен с помощью отчета.

**Объект PageFrame (Набор вкладок)**

Этот элемент управления представляет собой многостраничную форму, которая состоит из набора страниц, имеющих вкладки[[33]](#footnote-34) (рис. 159).

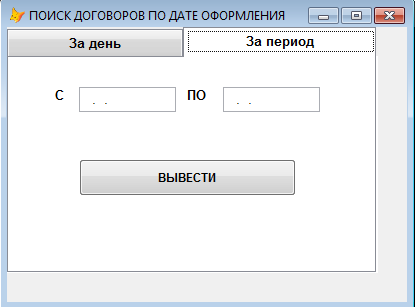


Рис. 159 – Пример формы с объектом PageFrame

Основные особенности работы с этим объектом вызваны тем, что это объект составного типа (подобно **Grid**): есть объект **PageFrame** в целом, а есть отдельные страницы (**Page**), входящие в его состав. Сам объект и его компоненты отображаются на панели **Properties**, *содержимое которой будет отличаться в зависимости от того, какая часть* ***PageFrame*** *выделена в списке выбора* (рис. 160).

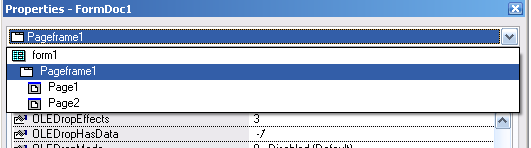


Рис. 160 – Объект **PageFrame** с двумя страницами в панели **Properties**

Основное свойство**PageFrame** – свойство **PageCount**, которое отвечает за количество страниц (вкладок) в группе. Форма может иметь до 99 страниц.

Основное свойство страницы **Page** – свойство **Caption**: надпись, которая появится в корешке страницы.

Чтобы получить доступ к отдельным составляющим **PageFrame**, можно выбрать нужный объект в панели **Properties** или перейти в режим редактирования **PageFrame**, вызвав на ней контекстное меню и выбрав в нем команду **Edit** (так же, как и при работе с **Grid**) – вокруг группы страниц появится голубая рамка выделения, а сами вкладки станут доступны для выбора страниц.

1. Запустите проект **dvd.pjx**.
2. Запустите конструктор форм.
3. Установите заголовок формы **Поиск договоров по дате оформления**.
4. Добавьте в источник данных формы **DataEnvironment** таблицу, в которой проводится поиск и из которой выводятся данные – **Prokat.dbf**.
5. Наложите на событие **Load** формы процедуру **Set Date German** (т.к. на форме предстоит работать с датами).
6. На панели **FormControls** выберите инструмент **PageFrame** (вкладки) и разместите его на форме протяжкой мыши так, чтобы вкладки заняли практически всю часть формы (рис. 161).

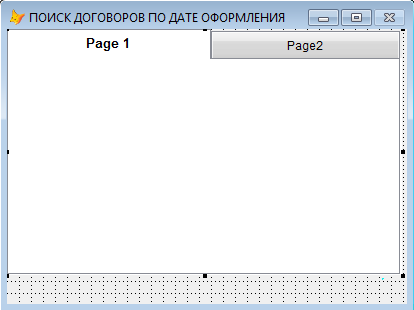


Рис. 161 – Размещение вкладок на форме

1. Не снимая выделения с **PageFrame**(в панели **Properties** отображается объект **Pageframe1**) установите значение свойства **PageCount = 3**. Обратите внимание, что количество вкладок увеличилось, добавилась одна вкладка справа.
2. Установите значение свойства **PageCount = 2**. Обратите внимание, что количество вкладок уменьшилось, удалена одна вкладка справа. Работа со свойством **PageCount** позволяет добавлять или удалять страницы в группу.
3. Измените заголовок первой страницы на**За день**следующим образом:
   1. вызовите на **PageFrame** контекстно меню и выберите команду **Edit**, сделав доступными вкладки по отдельности (обратите внимание, что вокруг **PageFrame** появилась голубая рамка выделения);
   2. выполните щелчок левой кнопкой мыши на первой вкладке**Page1**, обратите внимание, что на панели **Properties** в строке выбора объектов отображается **Page1**;
   3. на панели **Properties**измените свойство **Caption** = **За день** (обратите внимание, что изменилась подпись первой вкладки; при необходимости можно изменить шрифт, размер, начертание и прочие свойства, как для надписей **Label**).
4. Измените заголовок первой страницы на**За период** следующим образом:
   1. на панели **Properties** в строке выбора объекта выберите **Page2**;
   2. измените свойство **Caption** = **За период**(обратите внимание, что изменилась подпись второй вкладки).
5. На первой вкладке разместите тестовое поле для ввода даты и кнопку для запуска процедуры поиска:
   1. убедитесь, что **PageFrame** находится в режиме редактирования **Edit** (вкладки доступны для выделения по отдельности);
   2. выделите первую вкладку;
   3. выберите на панели **Form Controls** инструмент **TextBox** и разместите поле на первой вкладке;
   4. подобным образом разместите на первой вкладке кнопку с подписью**Вывести**;
   5. перейдите на вторую вкладку и убедитесь, что там не отображаются добавленные поле и кнопка (это своеобразный контроль правильности размещения объектов именно на первой вкладке).
6. Выполните щелчок левой кнопкой мыши в свободной области формы (в панели **Properties** в поле выбора объектов отображается **Pageframe1**). Попробуйте выделить добавленные поле и кнопку. Это тоже своеобразный контроль правильности выполнения задания – поле и кнопка располагаются на вкладке **Page1**, а не на **PageFrame** в целом.
7. Выберите первую вкладку и выделите добавленное текстовое поле. Вызовите на нем контекстное меню и выберите команду **Builder** для отображения построителя текстового поля. В списке **Data Type**установите тип данных **Date**.
8. Раскройте список выбора объектов панели **Properties**. Обратите внимание на структурно расположение объекта **Text1** (рис. 162) – поэтому приобращению к этому полю следует указать последовательно все ступени иерархии размещения: **thisform.Pageframe1.Page1.text1**. То же самое относится и к кнопке **Command1**.

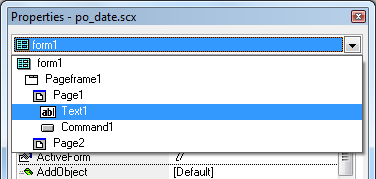


Рис. 162 – Отображение объектов вкладки в панели **Properties**

1. Для события **Click** кнопки**Вывести** разместите процедуру, запускающую отчет **dogovor** с использованием фильтрации по дате оформления:

\*Сохранение значения поля в переменную Х, чтобы сократить команды

Х=thisform.pageframe1.page1.text1.value

\*Вызов отчета с фильтрацией по дате оформления

REPORT FORM d:\dvd\reports\**dogovor.frx**FOR data\_pr=Х PREVIEW

1. Сохраните форму под именем **po\_date.scx** и запустите ее на выполнение. В поле ввода введите дату **26.10.2014** и нажмите кнопку**Вывести**. Обратите внимание на содержание отчета, перемещаясь по его страницам, – отображаются сведения только о тех договорах, которые были оформлены в указанную дату.
2. Поскольку результат поиска будет представлен несколько в другом виде, то создайте копию отчета **dogovor.frx** и добавьте его в проект:
   1. перейдите в папку **Reports** и создайте копии файлов **dogovor.frx** и **dogovor.frt**;
   2. присвойте копиям имена **dog\_data.frx**и **dog\_data.frt** соответственно;
   3. вернитесь в проект, выделите объект **Reports** и нажмите кнопку **Add**, в появляющемся окне отыщите файл **dog\_data.frx**.
3. Отредактируйте процедуру (п.14), изменив название отчета на **dog\_data.frx**.
4. Откройте для редактирования запрос **Kol\_Sum** (кнопка **Modify**), просмотрите SQL-код запроса и скопируйте его в буфер обмена.
5. Отредактируйте процедуру для события **Click** кнопки**Вывести** формы **po\_date.scx**, используя содержимое буфера обмена:[[34]](#footnote-35)

\*Сохранение значения поля в переменную Х, чтобы сократить команды

Х=thisform.pageframe1.page1.text1.value

\*подсчет количества и суммы по договорам проката за указанный день

SELECT COUNT(Prokat.kod\_pr),;

SUM(Prokat.summa\_zal+0.1\*Prokat.summa\_zal\*(Prokat.data\_dog-Prokat.data\_pr));

FROM ;

"dvd-bd!prokat";

WHERE Prokat.data\_pr = Х

\*Вызов отчета с фильтрацией по дате оформления

REPORT FORM d:\dvd\reports\**dog\_data.frx**FOR data\_pr=Х PREVIEW

1. Теперь необходимо добавить итоговые значения, рассчитанные с помощью запроса, в отчет. Для этого необходимо сохранить результат запроса в переменные. Т.к. в результате выполнения запроса получается два значения, то удобно сохранить их в массив из двух элементов. В дальнейшем элементы этого массива будут использованы в отчете, поэтому областью их действия должен быть весь проект (это повлияет на объявление массива). Областью действия переменной Х тоже должен быть весь проект, если требуется в отчет добавить дату поиска. Отредактируйте процедуру:

\*Сохранение значения поля в переменную Х, чтобы сократить команды

\*Областью действия Х станет весь проект

PUBLIC X

Х=thisform.pageframe1.page1.text1.value

\*Областью действия массива А станет весь проект

PublicarrayA(2)

\*подсчет количества и суммы по договорам проката за указанный день

SELECT COUNT(Prokat.kod\_pr),;

SUM(Prokat.summa\_zal+0.1\*Prokat.summa\_zal\*(Prokat.data\_dog-Prokat.data\_pr));

FROM ;

"dvd-bd!prokat";

WHERE Prokat.data\_pr = Х;

INTOARRAYA

\*Вызов отчета с фильтрацией по дате оформления

REPORT FORM d:\dvd\reports\**dog\_data.frx**FOR data\_pr=Х PREVIEW

1. Сохраните изменения в форме и запустите ее на выполнение. Введите в поле дату 26.10.2014 и нажмите кнопку**Вывести** – это действие выполнит запрос и проинициализирует массив **А**.
2. Завершите работу с формой и закройте отчет **dog\_data.**
3. Откройте отчет **Dog\_data** для редактирования в конструкторе, чтобы добавить в него поля с общим количеством, общей суммой и указанной в форме **po\_date.scx** датой (поля следует добавить в область **Title** (они отобразятся один раз на самой первой странице) или в область **PageHeader** (они будут отображаться один раз на каждой странице)):
   1. при необходимости отобразите панель инструментов ReportControls (ЗАДАНИЕ 23, п.14);
   2. на панели выберите инструмент **Field** (поле) и протяжкой создайте поле в области отчета **PageHeader**, что приведет к появлению окна **Field Properties**;
   3. нажмите кнопку  справа от поля Expression, что приведет к открытию окна построителя выражений **Expression Builder** (рис. 163);
   4. в списке **Variables** (переменные) отыщите переменную **а** (массив, в котором хранятся итоговые количество и сумма; во втором столбце указан тип переменной **A** (array, массив)) и выполните на ней двойной щелчок левой кнопкой мыши, чтобы добавить ее в поле **Expression**;

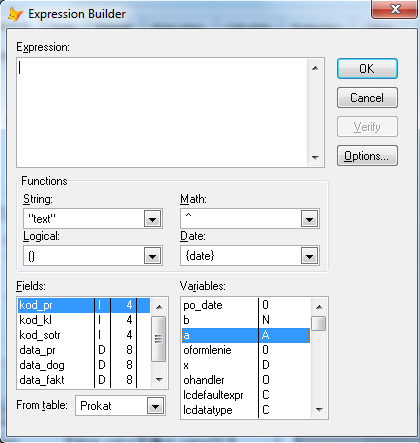


Рис. 163 – **Expression Builder** для добавления переменных в поля

* 1. отредактируйте выражение в поле **Expression** – **a(1)** – для добавления итогового количества, т.к. количество сохраняется в первом элементе массива**А**;
  2. нажмите кнопку ОК.

1. Действуя аналогично, добавьте в ту же область еще два поля – для отображения итоговой суммы (переменная **а(2)**) и даты поиска (переменная **х**).
2. Увеличьте размеры добавленных полей так, чтобы в них умещались подсчитанные значения; для поля с переменной х измените типа данных на **Date** и выполните настройки для отображения данных в формате дд.мм.гггг (ЗАДАНИЕ 23, п.7).
3. Закройте отчет, сохранив внесенные изменения.
4. Запустите форму **po\_date.scx**, введите в поле дату 26.10.2014 и нажмите кнопку**Вывести**. Убедитесь в появлении отчета с верными результатами.
5. Отредактируйте процедуру кнопки**Вывести** – отчет можно отобразить в том случае, когда соответствующие данные о договорах найдены:

\*Сохранение значения поля в переменную Х, чтобы сократить команды

\*Областью действия Х станет весь проект

PUBLIC X

Х=thisform.pageframe1.page1.text1.value

LOCATE FOR Prokat.data\_pr=X

IFFOUND()

\*Областью действия массива А станет весь проект

PublicarrayA(2)

\*подсчет количества и суммы по договорам проката за указанный день

SELECT COUNT(Prokat.kod\_pr),;

SUM(Prokat.summa\_zal+0.1\*Prokat.summa\_zal\*(Prokat.data\_dog-Prokat.data\_pr));

FROM ;

"dvd-bd!prokat";

WHERE Prokat.data\_pr = Х;

INTOARRAYA

\*Вызов отчета с фильтрацией по дате оформления

REPORT FORM d:\dvd\reports\**dog\_data.frx**FOR data\_pr=Х PREVIEW

ELSE

MESSAGEBOX(‘Договора не найдены’,16,’Сообщение’)

ENDIF

1. Снова откройте отчет в режиме конструктора и приведите его к виду, представленному на рис. 164 (при необходимости настройте формат отображения суммы).

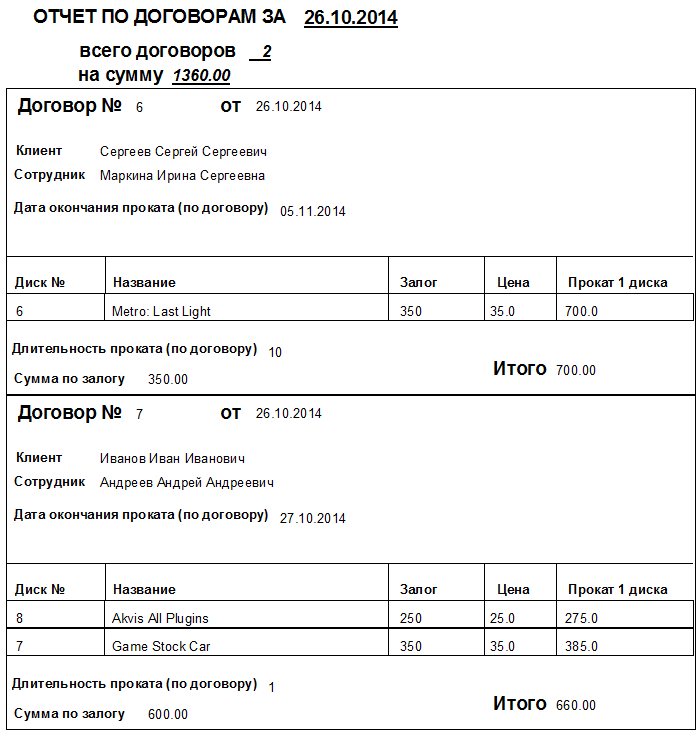


Рис. 164 – Вид отчета **Dog\_data**

1. Снова запустите форму и проверьте правильность формирования отчета за разные даты.
2. Откройте для редактирования в конструкторе форму **po\_date.scx** – необходимо доработать вторую вкладку.
3. Действуя аналогично п.10 – п.13, разместите на второй вкладке надписи, тестовые поля типа Date. Скопируйте и вставьте кнопку с первой вкладки и получите форму, представленную на рис. 165.

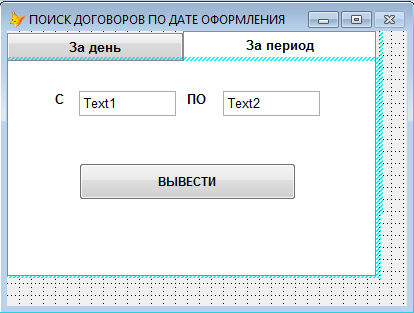


Рис. 165 – Вторая вкладка формы

1. Откройте процедуру кнопки**Вывести** второй вкладки формы и отредактируйте ее:

\*Сохранение значения полей в переменные Y и Z, чтобы сократить команды

\*Областью действия Y и Z станет весь проект

PUBLIC y,z

y=thisform.pageframe1.page2.text1.value

z=thisform.pageframe1.page2.text2.value

LOCATE FOR Prokat.data\_pr >= y AND Prokat.data\_pr <= z

IF FOUND()

\*Областью действия массива B станет весь проект

PUBLIC ARRAY B(2)

\*подсчет количества и суммы по договорам проката за указанный день

SELECT COUNT(Prokat.kod\_pr),;

SUM(Prokat.summa\_zal+0.1\* Prokat.summa\_zal\*(Prokat.data\_dog- Prokat.data\_pr));

FROM ;

"dvd-bd!prokat";

WHERE Prokat.data\_pr >= y AND Prokat.data\_pr <= z;

INTO ARRAY B

\*Вызов отчета с фильтрацией по дате оформления

REPORT FORM d:\dvd\reports\**dog\_period.frx[[35]](#footnote-36)**FOR data\_pr>=y AND data\_pr<=zPREVIEW

ELSE

MESSAGEBOX('Договора не найдены',16,'Сообщение')

ENDIF

1. Осталосьсоздатьотчет**dog\_period.frx**: впапке**Reports**создайтекопиифайлов**dog\_data.frx**и**dog\_data.frt**иприсвойтеимимена**dog\_period.frx**и**dog\_period.frt**соответственно.
2. Выделив в проекте объект **Reports**, нажав кнопку **Add**, добавьте к проекту отчет **dog\_period.frx**.
3. Приведите отчет к структуре, представленной на рис. 166.

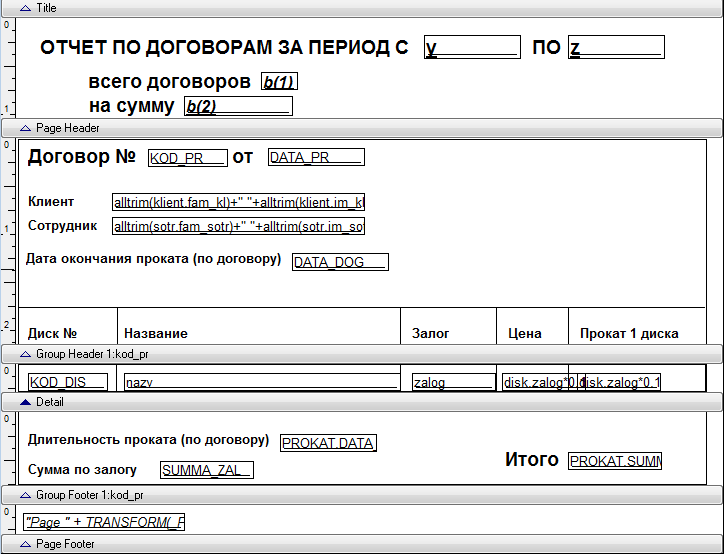


Рис. 166 – Макет отчета **dog\_period.frx**

1. Запустите форму **po\_date.scx**, перейдите на вторую вкладку, введите некоторый интервал дат (например, 26.10.2014 и 27.10.2014), нажмите кнопку Вывести и убедитесь в правильности формирования отчета-результата.
2. Завершите работу с проектом.

# 8. РАЗРАБОТКА МЕНЮ ПРИЛОЖЕНИЯ

## 8.1. Общие сведения о меню приложения. Структура меню

Итак, в проекте разработано большое количество объектов: база данных, формы, отчеты, запросы и представления, реализующие все функции будущего приложения, требуемые пользователем. Теперь необходимо объединить эти объекты и предложить пользователю *удобный* доступ ко всем возможностям приложения. Одним из вариантов такого объединения является меню, меню в привычном понимании – строка с командами, дающая доступ к функциям приложения.

Прежде чем приступить к созданию меню, следует определиться с его будущей структурой, командами и действиями, которые будут выполняться при запуске команд. Компоновка объектов и возможностей приложения зависит от разработчика.

Рис. 167 – Структура меню приложения

Пункт меню **Справочники** открывает формы для просмотра и редактирования информации о дисках, жанрах и категориях дисков, о клиентах и сотрудниках.

Пункт меню **Диски** содержит 4 подпункта:

* Каталог;
* В прокате;
* В наличии;
* Поиск.

Подпункт **Каталог** открывает отчет, отображающий информацию обо всех дисках проката, причем диски сгруппированы по жанрам и категориям, а также отсортированы по названиям. По каждому диску выводится информация о том, находится ли он в прокате или есть в наличии.

Подпункт**В прокате** открывает отчет, отображающий информацию обо всех дисках находящихся в прокате.

Подпункт**В наличии** открывает отчет, отображающий информацию обо всех дисках, которые есть в наличии.

Подпункт **Поиск**позволяет осуществить поиск дисков по критериям. Этот пункт содержит два подпункта:

* По году выпуска;
* По названию.

Подпункт**По году выпуска**открывает форму, реализующую поиск дисков, выпущенных не ранее указанного года. Результатом поиска является таблица, отображающая список найденных дисков, причем по каждому диску выводится информация о том, находится ли он в прокате или есть в наличии.

Подпункт**По названию** открывает форму, реализующую поиск дисков по части названия. Результатом поиска является таблица, отображающая список найденных дисков, причем по каждому диску выводится информация о том, находится ли он в прокате или есть в наличии.

Пункт меню **Договор** содержит 4 подпункта:

* Оформить;
* Править;
* Оформлены за период;
* Просрочены.

Подпункт меню**Оформить** открывает форму для оформления договора проката. При этом вводятся такие данные, как клиент, дата оформления проката, дата окончания проката, номер диска, автоматически рассчитывается длительность проката и общая стоимость.

Подпункт меню**Править** открывает форму, в которую вводится номер договора для редактирования. По введенному номеру договора открывается форма, в которую можно ввести фактическую дату окончания проката, рассчитать фактическую стоимость и определить сумму долга (DVD-проката перед клиентом или клиента перед DVD-прокатом).

Подпункт меню **Оформлены за период** служит для подведения итогов по оформленным прокатам за период или за один день.

Подпункт меню **Просрочены** позволяет вывести информацию о просроченных на сегодняшний день договорах проката. Этот подпункт содержит два подпункта:

* Кратко;
* Подробно.

Подпункт меню **Кратко** выводит форму, в которой отображается контактная информация о клиентах, которые не вернули диски вовремя.

Подпункт меню **Подробно** выводит форму, в которой отображается контактная информация о клиентах, которые не вернули диски вовремя, а также информация о дисках, которые были выданы по каждому просроченному договору.

Пункт меню **Справка** предоставляет краткие сведения о назначении команд меню.

Пункт меню **Выход** позволяет произвести выход из программы.

## 8.2. Подготовка к созданию меню

При создании меню будет необходимо прописывать команды для открытия форм, вывода отчетов, запуска запросов. С этими командами уже приходилось встречаться при выполнении заданий. Одна из особенностей любой команды вызова объекта проекта – это указание имени объекта и пути к нему. Но ведь папка DVD с объектами проекта не всегда может быть размещена на диске D:\. В этом случае есть возможность в начале работы с проектом указать папку, используемую по умолчанию (в данном случае, D:\DVD) и во всех остальных командах отпадет необходимость писать эту часть пути при обращении к объектам проекта. С этой целью используется команда:

**SET DEFAULT TO путь\_к\_папке\_по\_умолчанию**

В дальнейшем требуется отредактировать все объекты проекта, где использовались пути и имена фалов, чтобы удалить в них часть пути – папку по умолчанию.

### ЗАДАНИЕ 34. Настройка путей к объектам

1. Запустите проект **dvd.pjx**.
2. Вокне**Command**выполнитекоманду**Set default to D:\DVD.**
3. Откройте для редактирования в конструкторе форму **oformlenie.scx**.
4. Отобразите окно вида документа: меню **Tools** – **DocumentView** и выполните щелчок на форме.
5. В появившемся окне вида документа выделите строку**form1.buttonset1.cmdprint.click** – для отображения кода процедуры для события **Click** кнопки **Printreport**.
6. В окне кода процедуры отредактируйте команду вывода отчета, удалив из пути часть **D:\DVD**:

REPORT FORM reports\dogovor.frx FOR kod\_pr=thisform.kod\_pr1.value PREVIEW

1. Закройте форму, сохранив изменения.
2. Такие же изменение внесите в форму **redakt.scx**.
3. Откройте для редактирования форму **zapros.scx**.
4. В окне вида документа выделите строку **form1.command1.click** – для отображения коды процедуры для события **Click** кнопки**Найти**.
5. В окне кода процедуры отредактируйте команду открытия формы:

DO FORM forms\redakt.scx

1. Закройте форму, сохранив изменения.
2. Откройте для редактирования форму **po\_date.scx**.
3. В окне вида документа выделите строку **form1.pageframe1.page1.command1.click** – для отображения коды процедуры для события **Click** кнопки**Вывести**.
4. В окне кода процедуры отредактируйте команду просмотра отчета:

REPORT FORM reports\dog\_data.frx FOR data\_pr=x PREVIEW

1. В окне вида документа выделите строку **form1.pageframe1.page2.command1.click** – для отображения коды процедуры для события **Click** кнопки**Вывести**.
2. В окне кода процедуры отредактируйте команду просмотра отчета:

REPORT FORM reports\dog\_period.frx FOR data\_pr>=y AND data\_pr<=z PREVIEW

1. Закройте форму, сохранив изменения.
2. Завершите работу с проектом.

## 8.3. Создание справочной системы для пользователя

Разрабатываемое приложение направлено на использование. Поэтому должно быть понятно потенциальному пользователю. Если у пользователя возникнут вопросы по возможностям приложения, то он должен иметь возможность получить справочные сведения о них. Требуемую информацию пользователь может найти в руководстве пользователя или обратиться за консультацией к разработчику. Но проще всего воспользоваться справочной системой, содержащей информацию о приложении, описание его основных функций и инструкцию по работе.

Для отображения справочной информации в Visual FoxPro можно использовать строку состояния, всплывающие подсказки, диалоговые окна и справочную систему в формате Win Help или HTML.

В пособии не будет рассматриваться процедура создания справки в формате Win Help или HTML, т.к. для этого потребуются специализированные программы, а пособие ориентировано на применении Visual FoxPro. Справочная система будет реализована в виде формы с описанием возможностей команд меню.

Что касается диалоговых окон, то это некоторые сообщения для пользователя, возникающие, например, при некорректных действиях.

### ЗАДАНИЕ 35. Использование всплывающих подсказок и строки состояния

Строка состояния используется для пояснения назначения элементов управления формы и пунктов меню. Содержание строки состояния задается на панели **Properies** в свойстве **StatusBarText**. Кроме текста строки состояния вы можете задать текст подсказки, который будет отображаться под курсором мыши, если его установить на элемент управления и удерживать некоторое время. Текст подсказки задается в свойстве **TooiTipText**.

1. Запустите проект **dvd.pjx**.
2. Откройте для редактирования форму **oformlenie.scx**.
3. Выделите поле data\_pr1. На панели Properties измените значение свойств **TooiTipText = Введите дату выдачи дисков**, **StatusBarText = Поле ввода даты оформления договора (выдачи дисков)**.
4. Запустите форму.Наведите курсор мыши на поле **data\_pr1** и обратите внимание на всплывающую подсказку и содержимое статусной строки.
5. Завершите работу с формой.
6. Откройте для редактирования форму **redact.scx**. Создайте всплывающую подсказку для поля **data\_fakt1**: Введите дату возврата дисков и нажмите кнопку **Save record**.
7. Добавьте всплывающие подсказки к объектам других форм по вашему усмотрению.
8. Завершите работу с проектом.

### ЗАДАНИЕ 36. Разработка диалоговых окон

Разработчик может добавить в проект сообщения пользователю, возникающие при выполнении определенных действий. Набор сообщений зависит от фантазии разработчика. Для создания сообщений используется уже известная команда **Messagebox()**.

В результате выполнения задания будет разработаны сообщения, возникающее:

* при работе с формой поиска диска по названию в случае, если не заполнено поле ввода части названия диска;
* при оформлении договора проката дисков в случае, если дата возврата дисков меньше даты выдачи;
* при оформлении договора проката дисков в случае, если срок проката превышает 14 дней (при анализе предметной области было указано, что максимальный срок выдачи дисков составляет 14 дней).

1. Запустите проект **dvd.pjx**.
2. Откройте для редактирования форму**poisk\_nazv.scx**.
3. Выполните двойной щелчок на кнопке**Найти** для открытия окна кода процедуры для события **Click** кнопки.
4. Отредактируйте процедуру следующим образом:

IF EMPTY(thisform.text1.Value)

MESSAGEBOX('Поле не заполнено',16,'Предупреждение')

ELSE

\*обновление параметра поиска

thisform.text1.refresh

\*сохранить часть названия диска в переменную, удалив пробелы

x=ALLTRIM(thisform.text1.Value)

\*поиск в поле nazv без пробелов

LOCATE FOR ATCC(x,ALLTRIM(nazv))<>0

\*выполнить, если найдено

IF FOUND()

\*вывести, если найдено

BROWSE FIELDS nazv:H='Название':100, nal=IIF(v\_prok,'нет','да'):H='Наличие' FOR ATCC(x,ALLTRIM(nazv))<>0

ELSE

MESSAGEBOX('Диски не найдены',64,'Сообщение')

ENDIF

\*закрыть таблицу

ENDIF

1. Запустите форму и нажмите кнопку**Найти**, не вводя данные в поле. Убедитесь в формировании диалогового окна.
2. Откройте для редактирования форму **poisk\_god.scx**и отредактируйте процедуру для события **Click** кнопки**Найти** таким же образом (п.4).
3. Откройте для редактирования форму **oformlenie.scx** и выполните двойной щелчок левой кнопкой мыши по полу **data\_dog1** для отображения окна ввода кода процедуры. Убедитесь, что в списке **Object** отображается **data\_dog1**, в списке **Procedure** выберите событие **Valid**. Введите следующую процедуру:

IF this.Value<thisform.data\_pr1.Value

MESSAGEBOX(‘Данные не корректны’,16,'Предупреждение')

ENDIF

1. Запустите форму, перейдите к последней записи, перейдите в режим редактирования и исправьте дату возврата дисков на значение, меньшее даты выдачи. Нажмите кнопку**Сохранить** и убедитесь в формировании сообщения. Верните прежнее значение даты.
2. Отредактируйте созданную процедуру, добавив к ней строки:

IF this.Value-thisform.data\_pr1.Value>14

MESSAGEBOX(‘Срок проката не должен превышать 14 дней’,16,'Предупреждение')

ENDIF

1. Запустите форму, перейдите к последней записи, перейдите в режим редактирования и исправьте дату возврата дисков так, чтоб срок проката превысил 14 дней. Нажмите кнопку**Сохранить** и убедитесь в формировании сообщения. Верните прежнее значение даты.
2. Можете добавить диалоговые окна к другим формам по вашему усмотрению.
3. Завершите работу с проектом.

### ЗАДАНИЕ 37. Создание формы справочных сведений

1. Запустите проект **dvd.pjx**.
2. Создайте в проекте форму с помощью конструктора (**New** – **NewForm**).
3. Разместите на форме надпись и набор вкладок (ЗАДАНИЕ 34) так, чтобы она приняла вид, представленный на рис. 168.

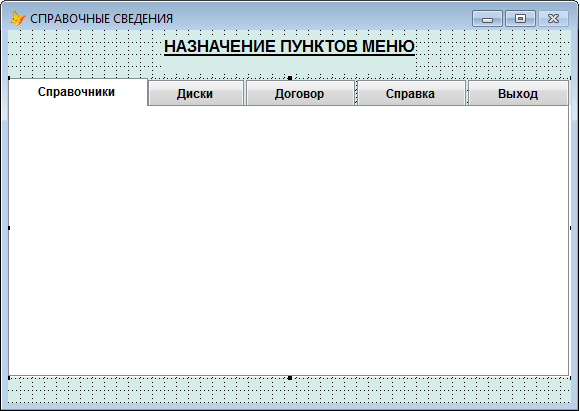


Рис. 168 – Вид формы **spravka.scx** в конструкторе

1. Используя инструмент **Label** и его свойства, разместите на вкладках информацию, как представлено на рис. 169 – 173.

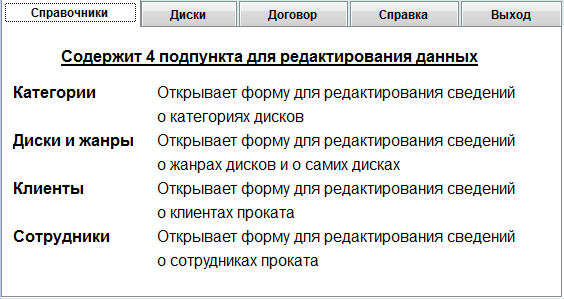


Рис. 169 – Содержимое вкладки **Справочники** формы **spravka.scx**

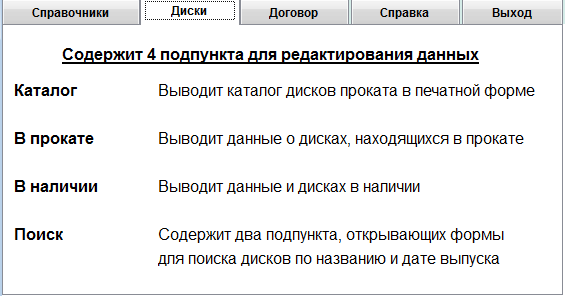


Рис. 170 – Содержимое вкладки **Диски** формы **spravka.scx**

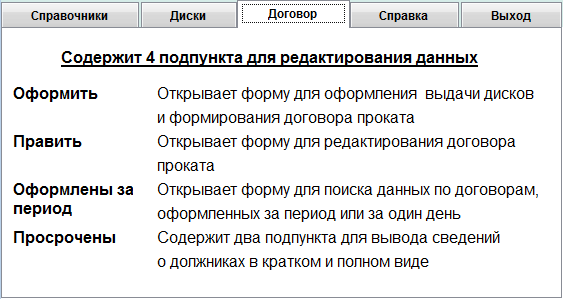


Рис. 171 – Содержимое вкладки **Договор** формы **spravka.scx**

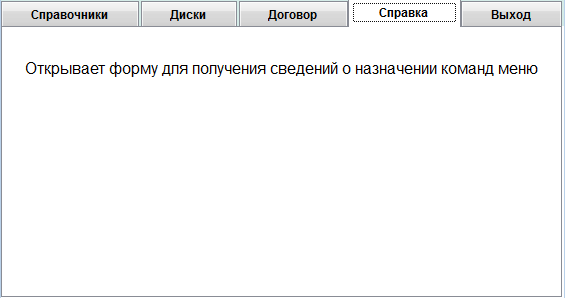


Рис. 172 – Содержимое вкладки **Справка** формы **spravka.scx**

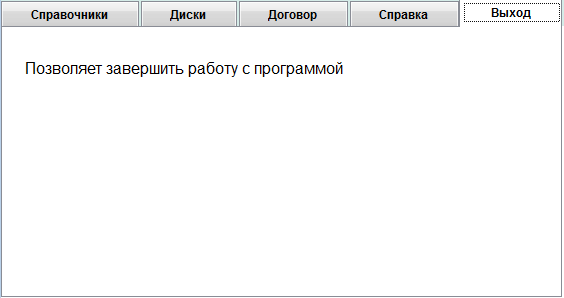


Рис. 173 – Содержимое вкладки **Выход** формы **spravka.scx**

1. Сохраните форму под именем **spravka.scx** и проверьте ее работоспособность.
2. Завершите работу с проектом.

## 8.4. Создание меню с помощью конструктора меню MenuDesigner

Создание меню, как и других объектов проекта, выполняется с помощью кнопки **New** при выделенном объекте **Menus** на вкладке **Other**.

VisualFoxPro позволяет создать меню двух типов:

* **Menu** – горизонтальное меню, расположенной в верхней строке экрана.
* **Shortcut** – всплывающее меню, в котором основные пункты расположены по вертикали.

Окно конструктора меню (каждого типа) содержит несколько управляющих столбцов (рис. 174):

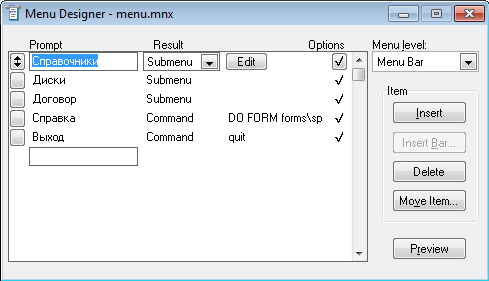


Рис. 174 –Окно конструктора меню

**Prompt** вводится название пункта меню или команды. Название задает пользователь.

**Result** задается тип действия. Можно организовать подменю (**Submenu**); задать выполняемую при выборе пункта меню процедуру (**Procedure**); определить команду, выполняемую при выборе пункта меню (**Command**); создать пункт, при выборе которого никакие действия выполняться не будут (**PadName**), пункт.используется для пояснения к другим пунктам меню.

**Options** задаются клавиш быстрого вызова пункта меню и некоторые другие параметры.

Вообще говоря, можно создавать меню любой вложенности и любого уровня, но лучше не увлекаться большим количеством уровней подменю.

В списке **Menulevel** (Уровень меню) указывается уровень текущего меню.

Слева в конструкторе меню размещены кнопки, назначение которых описано в табл. 17.

Таблица 17. Назначение кнопок конструктора меню

|  |  |
| --- | --- |
| **Кнопка** | **Назначение** |
| **Insert** (Вставить) | Добавляет в меню новый пункт |
| **InsertBar** (Вставить команды системного меню) | Открывает диалоговое окно **InsertSystemMenuBar**, содержащее команды системного меню Visual FoxPro, позволяя разместить их в создаваемом пользовательском меню |
| **Delete** (Удалить) | Удаляет текущий пункт меню |
| **MoveItem** (Переместить элемент) | Открывает одноименное диалоговое окно, позволяющее указать пункт меню, в который переносится текущий подпункт |
| **Preview** (Просмотр) | Размещает создаваемое меню на экране, позволяя просмотреть его внешний вид |

### ЗАДАНИЕ 38. Создание меню приложения

В результате выполнения задания будет разработано меню в виде горизонтальной строки. Работка конструктора такого типа меню аналогична работе конструктора всплывающего меню

1. Запустите проект **dvd.pjx**.
2. В окне Command выполните команду **SetdefaulttoD:\DVD**, чтоб установить папку **DVD** в качестве рабочего каталога по умолчанию.
3. В менеджере проекта перейдите на вкладку **Other** и нажмите кнопку New. В появляющемся окне запроса типа меню выберите вариант **Menu**, в результате чего на экране отобразится окно конструктора меню (рис. 175); обратите внимание на список **MenuLevel** – здесь отображается уровень редактируемого меню – **MenuBar** (строка меню).

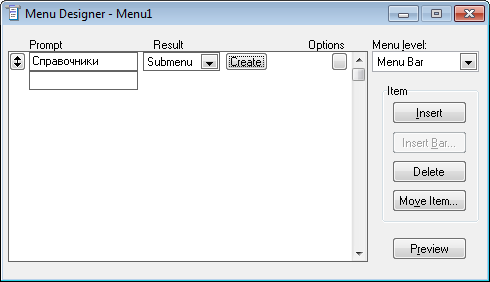


Рис. 175 – Окно конструктора меню

1. Создайте первый пункт меню **Справочники**:
   1. в столбце **Prompt** введите название пункта – **Справочники**;
   2. в столбце **Result** выберите вариант **Submenu** (т.к. пункт **Справочники** содержит четыре подпункта);
   3. нажмите кнопку в столбце **Options**, в результате чего будет открыто окно **PromptOptions** (рис. 176); установите курсор в поле **KeyLabel** (для задания комбинации клавиш для быстрого вызова пункта меню) и нажмите на клавиатуре комбинацию клавиш **Ctrl+Q**; убедитесь, что эта комбинация отобразилась в полях **KeyLabel** и **KeyText**;[[36]](#footnote-37)

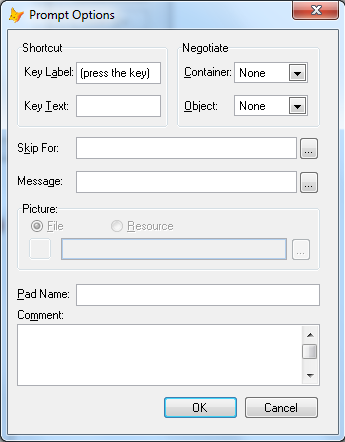


Рис. 176 – Окно **Prompt Options**

* 1. нажмите кнопку **Create**, в результате чего внешний вид конструктора изменится – теперь это конструктор создания подменю **Справочники** (рис. 177); обратите внимание на список **MenuLevel** – здесь отображается редактируемый уровень меню – **Справочники**;

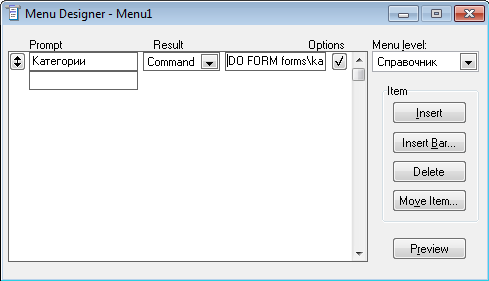


Рис. 177 – Окно конструктора пункта **Справочники**

* 1. в столбце **Prompt** введите название подпункта – **Категории;**
  2. в столбце **Result** выберите вариант **Command** (т.к. пункт будет использоваться для вызова формы **Kategor.scx**);
  3. в следующем столбце введите команду запуска формы **DO FORM forms\kategor.scx**;[[37]](#footnote-38)
  4. нажмите кнопку **Options** и задайте комбинацию клавиш для быстрого вызова пункта меню **Ctrl+W**(п.4-в);
  5. действуя аналогично п.4-д – п.4-з, создайте в подменю справочники еще три подпункта, описание которых дано в табл. 18; окно конструктора пункта **Справочники** примет вид, представленный на рис. 178;

Таблица 18. Описание команд пункта **Справочники**

|  |  |  |  |
| --- | --- | --- | --- |
| **Prompt** | **Result** | **Команда** | **Комбинация клавиш** |
| Диски и жанры | Command | DOFORMforms\ganr-disk.scx | Ctrl+E |
| Клиенты | Command | DOFORMforms\klient.scx | Ctrl+R |
| Сотрудники | Command | DO FORM forms\sotr.scx | Ctrl+T |

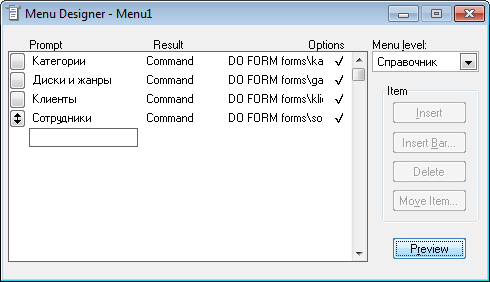


Рис. 178 – Окно конструктора пункта **Справочники** после  
добавления всех подпунктов

* 1. в окне конструктора нажмите кнопку **Preview** для предварительного просмотра меню, в результате чего на экране отобразится строка меню из одного пункта **Справочники**: раскройте его и убедитесь, что в нем отображаются созданные подпункты, выберите подпункт **Клиенты**; так же появилось окно параметров предпросмотра, в котором отображаются сведения о выбранном подпункте (рис. 179);

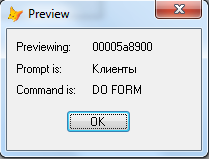


Рис. 179 – Окно параметров предпросмотра

* 1. нажмите в окне **Preview** кнопку ОК для выхода из режима предпросмотра меню.

1. В окне конструктора пункта в списке **Menu Level** выберите вариант **Menu Bar** для возврата к конструктору строки меню (рис. 175).
2. Действуя аналогично п.4, создайте остальные пункты и подпункты меню, руководствуясь схемой меню (рис. 167) и табл. 19. Периодически выполняйте предпросмотр меню, чтобы контролировать правильность его создания.

Таблица 19. Описание пунктов и подпунктов меню

|  |  |  |  |
| --- | --- | --- | --- |
| **Prompt** | **Result** | **Команда** | **Комбинация клавиш** |
| **Диски** | Submenu |  | Ctrl+Y |
| Каталог | Command | REPORT FORM reports\katalog.frx PREVIEW | Ctrl+U |
| Впрокате | Command | REPORT FORM reports\katalog.frx FOR v\_prok PREVIEW | Ctrl+I |
| В наличии | Command | REPORT FORM reports\katalog.frx FOR not v\_prok PREVIEW | Ctrl+О |
| Поиск | Submenu |  | Ctrl+А |
| По году выпуска | Command | DO FORM forms\poisk\_god.scx | Ctrl+S |
| По названию | Command | DO FORM forms\poisk\_nazv.scx | Ctrl+D |
| **Договор** | Submenu |  | Ctrl+F |
| Оформить | Command | DO FORM forms\oformlenie.scx | Ctrl+G |
| Править | Command | DO FORM forms\zapros.scx | Ctrl+H |
| Оформлены за период | Command | DO FORM forms\po\_date.scx | Ctrl+K |
| Просрочены | Submenu |  | Ctrl+L |
| Кратко | Command | DO FORM forms\dolgnik.scx | Ctrl+Z |
| Подробно | Command | DO FORM forms\dolgnik-1.scx | Ctrl+X |
| **Справка** | Command | DO FORM forms\spravka.scx | F1 |
| **Выход** | Command | QUIT | Ctrl+C |

1. Закройте конструктор меню, в результате чего на экране появится запрос на его сохранение, в котором нажмите кнопку **Yes** и сохраните меню под именем menu в папке **DVD**. Обратите внимание, что в менеджере проекта на вкладке **Other** в узле **Menus** появилось созданное меню **Menu**.
2. Выделите созданное меню и нажмите кнопку **Run**. Обратите внимание на отображение созданного меню в верхней строке окна (рис. 180). В строке отображается один пункт из стандартного меню FoxPro – после компиляции приложения этот недостаток будет устранен.



Рис. 180 – Разработанная строка меню

1. Выбирая последовательно каждый пункт и подпункт меню, убедитесь в его работоспособности. В последнюю очередь выберите пункт меню **Выход** и завершите работу с проектом.

# 9. РАЗРАБОТКА ОСНОВНОЙ ПРОГРАММЫ ПРОЕКТА И КОМПИЛЯЦИЯ ПРИЛОЖЕНИЯ

Проект обязательно должен содержать программу, которая запускает приложение и управляет его работой. Такой файл называется **основной программой** и им чаще всего является меню приложения. Вы можете также определить в качестве основной программы созданную вами программу управления приложением, содержащую установку основных параметров, используемых глобальных переменных и т. д.

Следует определить с тем, что из себя будет представлять основная программа для разрабатываемого приложения. Вообще говоря, возможно создать программу, в которой будут выполнены команды настроек FoxPro и команда, запускающая меню проекта. Но было бы неплохо при запуске приложения выполнить идентификацию пользователя и предоставить возможность работать с приложением только допущенным пользователям. В результате авторизации пользователя и будет запущено меню приложения. Итак, содержимое основной программы определено.

После разработки этой программы необходимо в проекте сделать ее главной (основной) – это будет необходимо для дальнейшей компиляции приложения и создания исполнимого файла.

В результате компиляции приложения для него будет разработан файл с расширением .exe, который и будет запускать главную программу.

### ЗАДАНИЕ 39. Разработка формы идентификации пользователя

В результате выполнения задания будет разработана форма идентификации пользователя, имя которого **Менеджер**, пароль **1234**.

1. Запустите проект **dvd.pjx**.
2. В окне **Command** выполните команду **SetdefaulttoD:\DVD** для задания рабочей папки по умолчанию.
3. Создайте с помощью конструктора форму **identif.scx**, структура которой приведена на рис. 181.

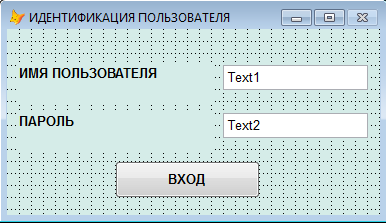


Рис. 181 – Структура формы идентификации пользователя

1. Для события **Click** кнопки **Вход** создайте процедуру:

if alltrim(thisform.text1.value)='Менеджер' AND (thisform.text2.Value)='1234' then

DO menu.mpr

thisform.Release

ELSE

MESSAGEBOX('Неверное имя пользователя и/или пароль',16, 'Ошибка')

endif

1. Закройте конструктор форм, в появляющемся окне запроса сохранения нажмите кнопку **Yes**и сохраните форму в папке **Forms** под именем **identif.scx**.
2. Запустите созданную форму и введите в поля ***неверные*** данные. Обратите внимание, что при заполнении поля пароля видны символы, образующие пароль. В реальности в подобных формах пароль при вводе заменяется символами, например **\***.
3. Закройте форму и откройте ее для редактирования в конструкторе.
4. Выделите поле **Text2** и измените значение свойства **PasswordChar = \***.
5. Запустите форму повторно, введите корректные имя пользователя и пароль, обратите внимание на поле ввода пароля.
6. Нажмите кнопку **Вход** и убедитесь в появлении меню.
7. Выберите пункт меню **Выход** и завершите работу с проектом.

### ЗАДАНИЕ 40. Разработка основной программы проекта

В результате выполнения задания будет разработан некоторый стандартный вариант основной программы, который можно использовать при разработке подобных приложений.

1. Запустите проект **dvd.pjx**.
2. Перейдите на вкладку **Code** и обратите внимание, что в узле объект **disk-2014** выделен жирным цветом – именно так и отмечается главная программа проекта.
3. Поскольку программа **disk-2014** не должна быть основной, то следует снять это выделение: вызовите контекстное меню на этом объекте и снимите флажок **Setmain**. Обратите внимание, что изменился внешний вид объекта.
4. Выделите объект**Programs**и нажмите кнопку **New** для вызова окна редактирования текста будущей основной программы.
5. Введите текст программы:

\*Настройки

SET TALK OFF[[38]](#footnote-39)

SET DATE GERMAN[[39]](#footnote-40)

SET CENTURY ON[[40]](#footnote-41)

SET ESCAPE OFF[[41]](#footnote-42)

SET DELETED ON[[42]](#footnote-43)

SET EXCLUSIVE OFF[[43]](#footnote-44)

SET DEFAULT TO "D:\dvd"[[44]](#footnote-45)

SET SAFETY OFF[[45]](#footnote-46)

ON SHUTDOWN quit[[46]](#footnote-47)

PUSH MENU \_msysmenu[[47]](#footnote-48)

DO FORM "forms\identif.scx"[[48]](#footnote-49)

\_screen.visible=.T.[[49]](#footnote-50)

\_screen.Caption='DVD-ПРОКАТ'[[50]](#footnote-51)

READ events

\*Восстановлениенастроек

POP MENU \_msysmenu

ON SHUTDOWN

\*Завершениепрограммы

CLEAR EVENTS

RETURN 0

1. Закройте окно программирования, в появляющемся запросе сохранения нажмите кнопку **Yes**и сохраните программу в папке **Programs** под именем **osnova**.
2. В окне проекта выделите программу **osnova**, вызовите на ней контекстное меню и включите флажок **Setmain**. Убедитесь, что в окне проекта программа выделена жирным шрифтом - теперь это основная программа приложения.
3. Запустите созданную программу кнопкой **Run**, в результате чего откроется форма идентификации пользователя.
4. В форме введите корректные данные и нажмите кнопку **Вход**, в результате чего на экране будет запущено меню приложения.
5. Можете опробовать пункты меню. В последнюю очередь выберите пункт меню **Выход** и завершите работу с проектом.

### ЗАДАНИЕ 41. Компиляция приложения

Итак, все компоненты приложения созданы. Проверена работоспособность объектов и не выявлено видимых ошибок. Теперь можно создать исполнимый файл, чтобы пользователю для работы с приложением не пришлось запускать FoxPro, ориентироваться в структуре проекта, искать в нем основную программу и осуществлять ее запуск. В идеале пользователю необходимо иметь только значок приложения. Который позволит выполнить его запуск.

Компиляция проекта выполняется при помощи кнопку **Build** окна проекта. В результате будет открыто окно настроек компиляции (рис. 182).

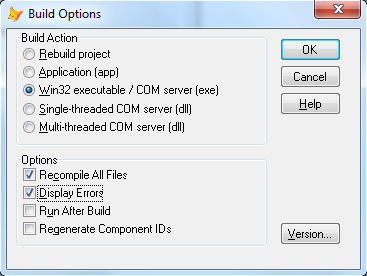


Рис. 182 – Окно **Build Options**

Окно позволяет:

* перестроить проект (**Rebuild project**);
* построить исполнимое приложение (**Application (app)**);
* построить приложение Win32 или COM-сервер (**Win32 executable/COM-server (exe)**);
* построитьдвавидабиблиотек (**Single-threaded COM server**или**Multi-threaded COM server**).

Чтобы построить приложение в виде **исполнимого файла**, следует выбрать вариант **Win32 executable/COM-server (exe)**. При этом в окне проекта должна быть выделена основная программа.

В этом же окне присутствуют флажки настройки режимов:

* **Recompile All Files** (Перекомпилировать все файлы) – компилирует заново все файлы проекта;
* **Display Errors** (Показать ошибки) – отображает возникшие в процессе компиляции ошибки;
* **Run After Build** (Запустить после построения) – по окончании компиляции проекта запускает полученное приложение на выполнение;
* **Regenerate Component Ids** (Перегенерировать идентификаторы компонентов) – осуществляет повторную генерацию идентификаторов каждого из компонентов проекта.

1. Запустите проект **dvd.pjx**.
2. В окне проекта на вкладке Programs выделите основную программу проекта **osnova** и нажмите кнопку **Build**.
3. Выполните настройки компиляции, как показано на рис. 182 и нажмите кнопку **ОК**. В результате будет произведено создание исполнимого файла приложения. Если все задания по разработке объектов проекта были выполнены правильно, то не должно возникнуть сообщений об ошибке.
4. Результатом создания станет появление запроса на сохранение исполнимого файла. Сохраните его под именем **dvd-prokat** в папке **DVD**.
5. Дождитесь появления в строке состояния сообщения **Buildexecutablecompleted** и завершите работу с проектом.
6. Откройте папку DVD и убедитесь в появлении в ней указанного исполнимого файла.
7. Запустите его на выполнение, в результате чего на экране должно появиться окно идентификации пользователя. Обратите внимание на заголовок окна – в нем отображается название созданного приложения. Обратите внимание на содержимое окна – здесь нет стандартных элементов FoxPro – панелей, окна проекта и др.
8. В окне идентификации пользователя введите корректные данные, чтоб запустить меню.
9. Опробуйте команды меню и убедитесь в работоспособности приложения.
10. Наверняка вы отметили одну особенность запускаемых с помощью меню форм – отсутствие изображения на кнопках перехода по записям (рис. 183).



Рис. 183 – Панель кнопок скомпилированного приложения

1. Перейдите в папку с FoxPro – обычно **C:\ProgramFiles\MicrosoftVisualFoxPro 9** – и найдите в ней папку **Wizards**. Скопируйте ее в буфер обмена.
2. Перейдите в рабочую папку **D:\DVD** и вставьте в нее содержимое буфера обмена.
3. Снова запустите проект **dvd.pjx**.
4. Теперь откройте для редактирования любую форму, содержащую кнопки перехода по записям, например **Klient.scx**.
5. Выделите панель кнопок, вызовите на ней контекстное меню и выберите команду **Edit** (чтобы перейти в режим редактирования панели).
6. Выделите первую кнопку – переход к первой записи. Отыщите в панели **Properties** свойство **Picture** и отредактируйте его содержимое, удалив часть свойства:

Исходное значение **=HOME()+"WIZARDS\WIZBMPS\WZTOP.BMP"**

Итоговое значение **WIZARDS\WIZBMPS\WZTOP.BMP**

1. Подобным образом измените свойство **Picture**следующих трех кнопок перехода по записям.
2. Такие же изменения необходимо внести в кнопки и на остальных формах. Чтобы не выполнять эти настройки многократно, можно скопировать панель кнопок из текущей формы и разместить копию на остальных формах*. Единственное, что следует сделать в копии – это добавить процедуры, связанные с прежней панелью, и выполнить настройки кнопок, существующие на прежней панели*.
3. Выделите отредактированную панель кнопок полностью (не должно быть голубой рамки выделения, а только ограничивающие черные маркеры) и скопируйте ее в буфер обмена.
4. Закройте форму, сохранив изменения.
5. Удобнее сначала скопировать панель кнопок в те формы, в которых их не придется редактировать – это все формы проекта, кроме **Dolgnik-1.scx**, **Oformlenie.scx**, **Redakt.scx**. Открывайте поочередно все формы проекта (содержащие панель кнопок) и вставляйте из буфера обмена панель с новыми настройками, а прежние панели кнопок просто удалите.
6. Откройте в конструкторе форму**Dolgnik-1.scx** – в ней часть кнопок были скрытыми. Вставьте в нее содержимое буфера обмена – новую панель кнопок – и скройте кнопки поиска, печати, добавления, редактирования и удаления записей (ЗАДАНИЕ 32, п.27-в). Закройте форму.
7. Откройте в конструкторе форму **Oformlenie.scx** – в ней с кнопкой **Printreport** была связана процедура. Вставьте на форму содержимое буфера обмена – новую панель.
8. Выделите прежнюю панель и перейдите в режим ее редактирования, вызвав на ней контекстное меню и выбрав команду Edit. Выделите кнопку **Printreport**и выполните на ней двойной щелчок левой кнопкой мыши для открытия окна соответствующей процедуры.
9. Выделите процедуру и скопируйте ее в буфер обмена.
10. Повторяя действия п.23, откройте окно ввода процедуры для события Click кнопки **Printreport** новой панели кнопок.
11. Вставьте процедуру из буфера обмена.
12. Удалите прежнюю панель кнопок, а новую панель скопируйте в буфер обмена.
13. Закройте форму.
14. Откройте для редактирования форму **Redakt.scx** и вставьте в нее панель кнопок из буфера обмена, а прежнюю панель просто удалите, т.к. они полностью идентичны. Закройте форму.
15. Повторяя действия п.2 – п.9, перекомпилируйте проект, завершите работу с FoxPro, запустите приложение с помощью исполнимого файла и опробуйте команды меню приложения. Убедитесь в корректности отображения кнопок в формах.
16. Используя команду меню **Выход**, завершите работу с приложением.

# ПРИЛОЖЕНИЕ 1 Некоторые сообщения об ошибках и их описание

При создании форм (отчетов) с помощью мастера на первом шаге происходит выбор полей таблиц базы данных, помещаемых в форму. И достаточно часто в соответствующем окне мастера отсутствуют таблицы для выбора (рис. 184). В этом случае следует нажать кнопку  справа от списка **FreeTables** и в открывшемся окне найти файл с таблицей базы данных, на основе которой создается форма (отчет).

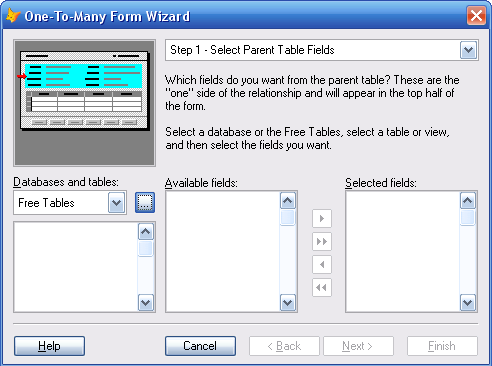


Рис. 184 – Первый шаг мастера форм и отсутствие в нем таблицы для выбора полей

При открытии проекта, при запуске его объектов или открытии их для редактирования при переносе проекта с одного компьютера на другой может появиться сообщение об отсутствии файла класса (рис. 185). В таком случае следует нажать в окне кнопку **Locate** и найти требуемый файл в папке с VisualFoxPro (рис. 186) – что это за файл и где его искать, указано в самом сообщении.

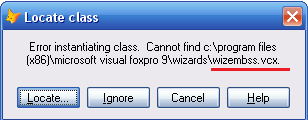


Рис. 185 – Сообщение о невозможности найти файл класса объектов

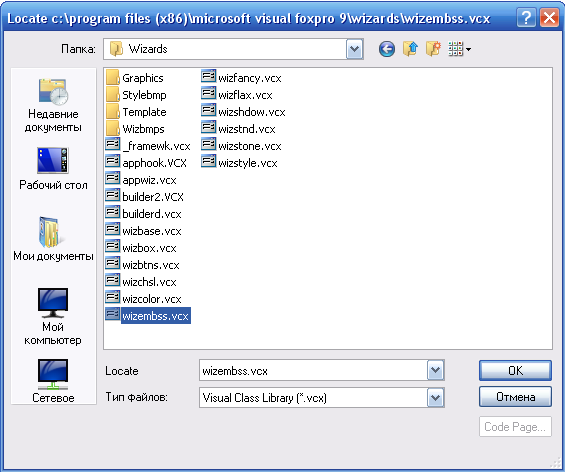


Рис. 186 – Указание требуемого файла

При запуске некоторых объектов проекта может появиться сообщение о невозможности найти базу данных (рис. 187). В таком случае следует нажать в окне кнопку **Locate** и найти файл базы данных самостоятельно.

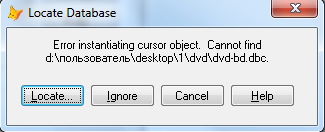


Рис. 187 – Сообщение о невозможности найти файл базы данных

При запуске некоторых объектов проекта может появиться сообщение о невозможности использовать таблицу базы данных (рис. 188). В таком случае в окне Command следует выполнить команду Use. Если это не снимает проблемы, то в окне Command следует выполнить команду CloseTables. Если и это не решает проблему, то следует закрыть и снова открыть проект.

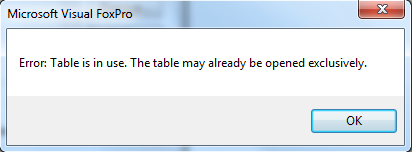


Рис. 188 – Сообщение об ошибке открытия таблицы

Такое же сообщение может только появиться при проверке работоспособности скомпилированного приложения или при проверке работоспособности меню. Обычно такое сообщение возникает при попытке запуска формы или просмотра отчета и связано с тем, что на предыдущих этапах работы с программой была отображена таблица Browse(как результат запроса) или была реализована процедура поиска с использованием команды открытия таблицы базы данных с помощью команды USE.

Если в приложении разрабатывается форма поиска информации, то рекомендуется:

при организации поиска по полям одной таблицы добавить эту таблицу в DataEnvironmentформы и не использовать в процедуре поиска команду Use;

при организации поиска по полям нескольких таблиц создать представление, объединяющее нужные таблицы, и тогда форма поиска будет создана на основе одной таблицы-представления, которую и надо добавить в DataEnvironment, чтобы не использовать в процедуре поиска команду Use.

Если в приложении предполагается вывод результатов запросов, то эти результаты лучше оформить в виде формы или отчета.

При работе с источником данных при добавлении таблиц может появиться таблица, имя которой отличается от имени таблицы базы данных (рис. 189, при добавлении таблицы Sotr в схему данных добавлена таблица Sotr1). Значит, верная таблица тоже добавлена, но она просто не видна в схеме данных: скрыта под какой-то другой таблице или размер окна схемы данных мал. В этом случае стоит передвинуть таблицы в схеме данных, чтобы под ними увидеть нужную таблицу (в данном случае, Sotr). А лишнюю таблицу (в данном случае, Sotr1)следует удалить.

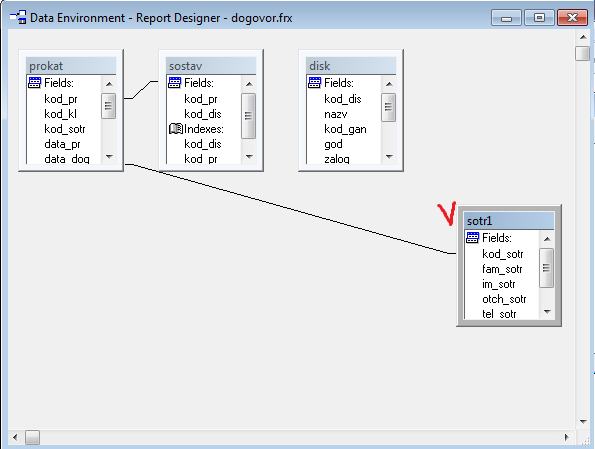


Рис. 189 – Окно источника данных с лишней таблицей

При вызове свойств отчета или его составляющих может появиться сообщение (рис. 190), говорящее о том, что нет подключенного принтера. Следует нажать в окне сообщения кнопку ОК и продолжить работу с отчетом.

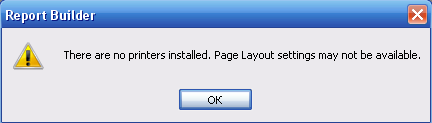


Рис. 190 – Сообщение при вызове свойств отчета или его составляющих

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Голицына о.л., Максимов Н.В., Попов И.И. Базы данных: Учеб.пособие. – 3-е изд., испр. и доп. – М.: ФОРУМ: ИНФРА-М, 2011.
2. Клепинин В.Б., Агафонова Т.П. Visual FoxPro 9.0. Наиболее полное руководство. – СПб.: БХВ-Петербург, 2008.
3. Лебедев А.Н. Visual FoxPro 9 / Лебедев А.Н. – М.: НТ Пресс, 2005.
4. Омельченко Л.Н. Самоучитель Visual Foxpro 8. – СПб.: БХВ-Петербург, 2003.
5. Проектирование автоматизированных информационных систем / Н.М. Абдикеев, Н.З. Емельянова, Т.Л. Партыка, В.П. Романов; Под ред. К.И. Курбакова. – М.: Издательский центр «Академия», 2010.

1. Будем считать, что подразумевается номер мобильного телефона в формате ххх-ххх-хх-хх. [↑](#footnote-ref-2)
2. Функция **atсс()** позволяет производить поиск символов в содержании поля. При успешном поиске функция возвращает ненулевое значение. [↑](#footnote-ref-3)
3. Формат для отображения числового значения без десятичных знаков. [↑](#footnote-ref-4)
4. Формат для отображения числового значения с двумя знаками после десятичной точки. [↑](#footnote-ref-5)
5. ALLTRIM() – удаление пробелов из строки. [↑](#footnote-ref-6)
6. ATCC() – поиск подстроки в строке. Функция возвращает значение ноль, если подстрока не найдена. [↑](#footnote-ref-7)
7. IIF(условие, выражение1, выражение2) – условная функция, в которой проверяется «условие» и используется «выражение1», если условие истинно, и «выражение2», если – ложно. [↑](#footnote-ref-8)
8. Не обращайте внимание на появление полей wizard1, wizard2 [↑](#footnote-ref-9)
9. Если созданное поле с ценой за диск отображается с десятичными знаками, то можно изменить формат поля в окне **FieldProperties**, если на вкладке **Format** в поле **FormatExpression** ввести шаблон **9999**, например. [↑](#footnote-ref-10)
10. Для ввода параметров **“Да”** и **”Нет”** можно использовать функцию **“text”** из группы **Logiсal**. [↑](#footnote-ref-11)
11. При создании выражений для вычислений используйте копирование существующих полей и их описаний. [↑](#footnote-ref-12)
12. **Structured Query Language** – структурированный язык запросов [↑](#footnote-ref-13)
13. В отличие от постоянных отношений, определяемых между таблицами в базе данных, при объединении таблиц в конструкторе запросов вы можете использовать любые поля таблиц. [↑](#footnote-ref-14)
14. Установленные между таблицами условия объединения можно изменять. Для этого необходимо использовать вкладку **Join** (Объединение) конструктора запросов. [↑](#footnote-ref-15)
15. По умолчанию Visual FoxPro при объединении таблиц использует опцию **Inner join** (Внутреннее объединение), при которой из таблиц выбираются только те записи, которые содержат совпадающие значения в полях связи. [↑](#footnote-ref-16)
16. Эта кнопка используется для отбора записей, не удовлетворяющих указанному условию. [↑](#footnote-ref-17)
17. Содержимое таблицы может отличаться. [↑](#footnote-ref-18)
18. При необходимости данную таблицу можно добавить к проекту к объектам **Free Tables**. Для этого в проекте на вкладке **Data** следует выбрать объект **Free Tables**, нажать кнопку **Add** и выбрать необходимую таблицу. [↑](#footnote-ref-19)
19. Содержимое в реальности может отличаться. Символ «;» используется для разделения строк: если его удалить, то разделяемые им строки должны быть написаны на одной строке. [↑](#footnote-ref-20)
20. В новых заголовках столбцов не допускается использовать пробелы. Лучше составлять их только из букв. [↑](#footnote-ref-21)
21. Обратите внимание на область **Select**: **Tables** или **Views**, т.е. источниками данных для представления могут служить как таблицы базы данных, так и представления (но не запросы) [↑](#footnote-ref-22)
22. Удобнее добавлять таблицы в область данных в определенном порядке: второй добавить таблицу, связанную с первой. [↑](#footnote-ref-23)
23. Обратите внимание, что конструктор представлений выглядит так же, как и конструктор запросов. [↑](#footnote-ref-24)
24. Обратите внимание, что окно SQL-кода представления (в отличие от запроса) помимо основной части содержит часть, отвечающую за свойства полей [↑](#footnote-ref-25)
25. Именно поле первичного ключа – единственное гарантированно непустое. [↑](#footnote-ref-26)
26. Функция ctod() преобразует аргумент в дату; дата записывается в том формате, который сейчас настроен (в нашем случае, дд.мм.гггг). [↑](#footnote-ref-27)
27. Не забудьте, что из отчета нельзя удалять используемые при итоговых вычислениях поля [TotalCount] и [CountFor]. [↑](#footnote-ref-28)
28. Укажите сегодняшнюю дату [↑](#footnote-ref-29)
29. Укажите сегодняшнюю дату [↑](#footnote-ref-30)
30. Из таблицы Prokat все поля, кроме kod\_sotr, из таблицы Klient все поля, кроме kod\_kl. [↑](#footnote-ref-31)
31. Функция Empty(поле) возвращает значение «истина», если поле пусто, и «ложно» - если поле не пусто. [↑](#footnote-ref-32)
32. Содержимое окна-результата может быть разным в зависимости от даты возврата в исходных договорах. При необходимости, можете их поменять, чтобы проверить работоспособность представления. [↑](#footnote-ref-33)
33. Вкладки, созданные с помощью инструмента **PageFrame**, подобны привычным вкладкам в диалоговых окнах Windows-программ. [↑](#footnote-ref-34)
34. Дата 26.10.2014 заменена на **Х** [↑](#footnote-ref-35)
35. Это отчет, в котором будут отображаться результаты поиска. [↑](#footnote-ref-36)
36. При необходимости можно задать сообщение, появляющееся в строке состояния при выборе пункта (**Message**), комментарий к пункту меню (**Comment**)и другие параметры. [↑](#footnote-ref-37)
37. Обратите внимание, на отсутствие в пути части **D:\DVD** и на название формы – оно должно совпадать с названием формы в проекте. Удобно запускать формы на выполнение, затем редактировать команды в окне **Command**и копировать их. [↑](#footnote-ref-38)
38. Определяет, будет ли Visual FoxPro отображает результаты выполнения команды. [↑](#footnote-ref-39)
39. Настройка формата дат. [↑](#footnote-ref-40)
40. Отображать четыре (а не две) цифры при указании года в датах. [↑](#footnote-ref-41)
41. Разрешить или запретить клавише ESC прерывать выполнение команд. [↑](#footnote-ref-42)
42. Определяет, будет ли Visual FoxPro обрабатывать записи, помеченные для удаления, и будут ли они доступны для использования с другими командами. [↑](#footnote-ref-43)
43. Определяет, будет ли Visual FoxPro открывает таблицу для исключительного или совместного использования в сети. [↑](#footnote-ref-44)
44. Установка рабочего каталога. ***Если папка с проектом будет перенесена в другой каталог, то достаточно выполнить изменение именно в этой строке программы и перекомпилировать проект (ЗАДАНИЕ 41)***. [↑](#footnote-ref-45)
45. Разрешить или запретить перезаписывать объекты без запроса. Используется при сохранении результатов запросов в таблицы. [↑](#footnote-ref-46)
46. Задает команду, которая выполняется при попытке выйти из Visual FoxPro или Windows. [↑](#footnote-ref-47)
47. Загрузка системного меню. [↑](#footnote-ref-48)
48. Запуск формы идентификации пользователя. [↑](#footnote-ref-49)
49. Отображение рабочего окна. [↑](#footnote-ref-50)
50. Содержание строки заголовка окна. [↑](#footnote-ref-51)