

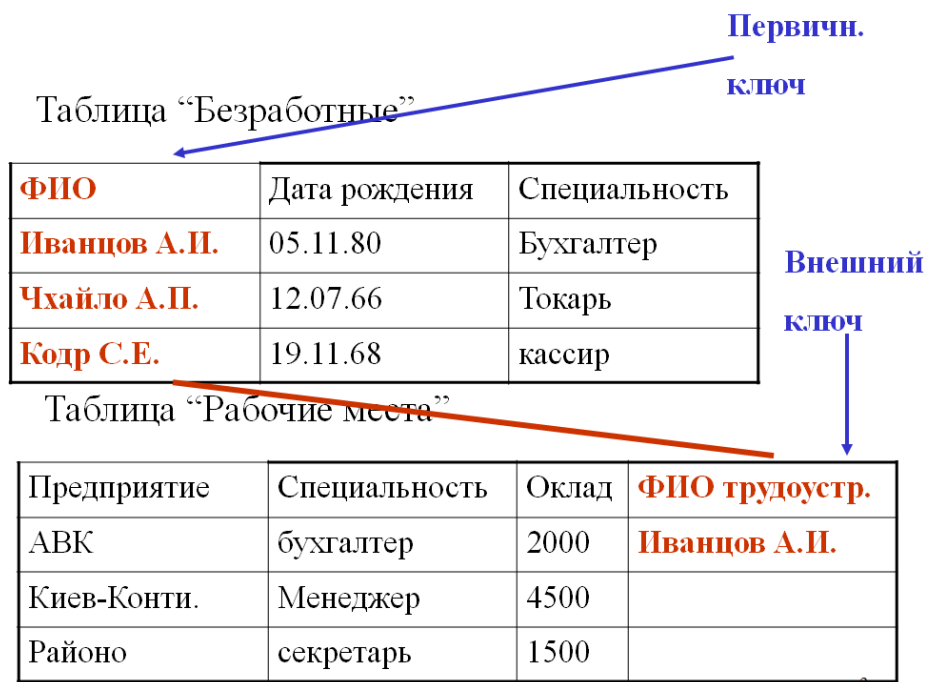
1. Первичный вторичный и внешний ключи. Их назначение.

Первичный ключ- это атрибут или набор атрибутов, однозначно характеризующий каждую запись таблицы

Внешний ключ- поле таблицы, предназначенное для хранения значения первичного ключа другой таблицы с целью организации связи между этими таблицами

Вторичный ключ - это одно или несколько полей (столбцов) в таблице, содержащих ссылку на поле или поля первичного ключа в другой таблице. Связь между таблицами устанавливается по совпадению в них значений двух полей- **первичного** и **внешнего** ключей. Из двух связанных таблиц одна является главной (таблица-отец), а другая- подчиненной (таблица-сын). Главная таблица содержит первичный ключ, а подчиненная- внешний. Если первичный ключ главной таблицы состоит из нескольких полей или представляет собой одно, но длинное поле, его заменяют коротким **искусственным ключом** (кодом, шифром и т.п.)

Это сокращает **суммарный** размер связанных таблиц.



1:1

Каждому значению первичного ключа в главной таблице соответствует не более одной записи в подчиненной таблице

1:M

Каждому значению первичного ключа в главной таблице соответствует не более одной записи в подчиненной таблице. Если при связи «один ко многим» первичный ключ достаточно длинный или состоит из нескольких полей, то в главную таблицу добавляют более короткий **искусственный ключ**, который используется в качестве внешнего ключа в подчиненной таблице.

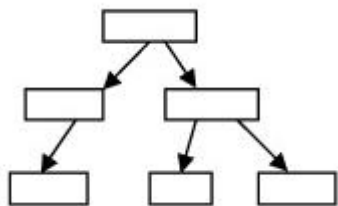
Размер главной таблицы увеличивается, однако суммарный объем двух связанных таблиц сокращается

2. Иерархическая, сетевая, реляционная и постреляционная модели данных.

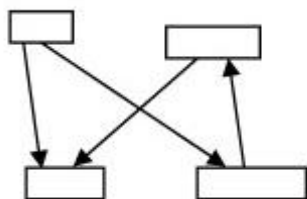
Иерархическая модель данных — это модель данных, где используется представление базы данных в виде древовидной (иерархической) структуры, состоящей из объектов (данных) различных уровней.

Между объектами существуют связи, каждый объект может включать в себя несколько объектов более низкого уровня. Такие объекты находятся в отношении предка (объект более близкий к корню) к потомку (объект более низкого уровня), при этом возможна ситуация, когда объект-предок не имеет

потомков или имеет их несколько, тогда как у объекта-потомка обязательно только один предок. Объекты, имеющие общего предка, называются близнецами (в программировании применительно к структуре данных дерево устоялось название братья).



Сетевая модель данных — логическая модель данных, являющаяся расширением иерархического подхода, строгая математическая теория, описывающая структурный аспект, аспект целостности и аспект обработки данных в сетевых базах данных.

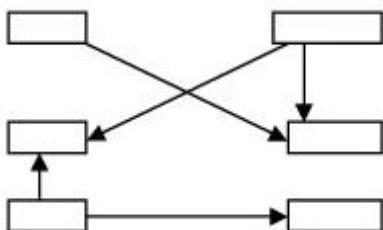


Реляционная модель данных (РМД) — логическая модель данных, прикладная теория построения баз данных, которая является приложением к задачам обработки данных таких разделов математики, как теория множеств и логика первого порядка.

На реляционной модели данных строятся реляционные базы данных.

Реляционная модель данных включает следующие компоненты:

- Структурный аспект (составляющая) — данные в базе данных представляют собой набор отношений.
- Аспект (составляющая) целостности — отношения (таблицы) отвечают определенным условиям целостности. РМД поддерживает декларативные ограничения целостности уровня домена (типа данных), уровня отношения и уровня базы данных.
- Аспект (составляющая) обработки (манипулирования) — РМД поддерживает операторы манипулирования отношениями (реляционная алгебра, реляционное исчисление).



Постреляционная модель данных представляет собой расширенную реляционную модель, снимающую ограничение неделимости данных, хранящихся в записях таблиц. Постреляционная модель данных допускает многозначные поля - поля, значения которых состоят из подзначений. Набор значений многозначных полей считается самостоятельной таблицей, встроенной в основную таблицу.

3. Состав индексного файла

Тип индексного файла	Описание	Содержит	Ограничения
Структурные составные индексные файлы Structural compound index	Автоматически открываются и закрываются при открытии и закрытии таблиц.	множество индексных ключей	в 240-символов на оцениваемом выражении

(.cdx)	Используются те же базовые имена, что и имена табличных файлов.
Неструктурные составные индексные файлы Nonstructural compound index (.cdx)	<p>Должен быть открыт эксклюзивно.</p> <p>Используются имена, отличное от базовых имен табличных файлов и определяются пользователем.</p> <p>множество в 240-символов индексных на оцениваемом ключей выражении</p>
Самостоятельные индексные файлы Standalone index (.idx) files	<p>Должен быть открыт эксклюзивно.</p> <p>Используются имена, отличное от базовых имен табличных файлов и определяются пользователем.</p> <p>Единичный в 100-символов индексный на оцениваемом ключ выражении</p>

При создании индекса, вы предоставляете выражение, которое содержит имена полей, определяющее порядок, в котором должны быть организованы записи. Visual FoxPro создает индексный ключ для каждой записи в таблице, основанный на определенном вами индексном выражении и хранит индексные ключи в индексных файлах. Индексный файл хранит и обрабатывает указатели на записи в табличном файле (.dbf) и организует их в соответствии со значениями индексных ключей. Индексный файл является отдельным файлом, но связан с табличным .dbf-файлом. Visual FoxPro поддерживает приведенные далее типы индексных файлов: структурный составной индексный файл (structural compound index (.cdx)), неструктурный составной индексный файл (nonstructural compound index (.cdx)), а также самостоятельные индексные файлы (standalone index (.idx)). Структурные и неструктурные .cdx-файлы содержат множественные индексы, которые имеют имена или тэги, которые идентифицируют их, тогда как самостоятельный индексный .idx-файл содержит только единичный индекс.

4. Недостатки файл-серверной модели базы данных.

По способу доступа к БД

Файл-серверные

В файл-серверных СУБД файлы данных располагаются централизованно на файл-сервере. СУБД располагается на каждом клиентском компьютере (рабочей станции). Доступ СУБД к данным осуществляется через локальную сеть. Синхронизация чтений и обновлений осуществляется посредством файловых блокировок. **Преимуществом** этой архитектуры является низкая нагрузка на процессор файлового сервера. **Недостатки:** потенциально высокая загрузка локальной сети; затруднённая или невозможность централизованного управления; затруднённая или невозможность обеспечения таких важных характеристик как высокая надёжность, высокая доступность и высокая безопасность. Применяются чаще всего в локальных приложениях, которые используют функции управления БД; в системах с низкой интенсивностью обработки данных и низкими пиковыми нагрузками на БД. На данный момент файл-серверная технология считается устаревшей, а её использование в крупных информационных системах — недостатком. Примеры: Microsoft Access, Paradox, dBase, FoxPro, Visual FoxPro

5. Представления данных

Представлением называется SQL-запрос на выборку, которому присвоили имя и который затем сохранили в базе данных. Представление позволяет пользователю увидеть результаты сохраненного

запроса, а SQL обеспечивает доступ к этим результатам таким образом, как если бы они были реальной таблицей базы данных. Представления используются по нескольким причинам: они позволяют сделать так, что разные пользователи базы данных будут видеть ее по-разному; их помощью можно ограничить доступ к данным, разрешая пользователям видеть только некоторые из строк и столбцов таблицы; они упрощают доступ к базе данных, показывая каждому пользователю структуру хранимых данных в наиболее подходящем для него виде. Преимущества представлений: Использование представлений в базах данных различных типов может оказаться полезным в самых разнообразных ситуациях. В базах данных на персональных компьютерах представления применяются для удобства и позволяют упрощать запросы к базе данных. В промышленных базах данных представления играют главную роль в создании собственной структуры базы данных для каждого пользователя и обеспечении ее безопасности. Основные преимущества представлений перечислены ниже.

- **Безопасность** Каждому пользователю можно разрешить доступ к небольшому числу представлений, содержащих только ту информацию, которую ему позволено знать. Таким образом можно осуществить ограничение доступа пользователей к хранимой информации.

- **Простота запросов** С помощью представления можно извлечь данные из нескольких таблиц и представить их как одну таблицу, превращая тем самым запрос ко многим таблицам в однотабличный запрос к представлению.

- **Структурная простота** С помощью представлений для каждого пользователя можно создать собственную структуру базы данных, определив ее как множество доступных пользователю виртуальных таблиц.

- **Защита от изменений** Представление может возвращать непротиворечивый и неизменный образ структуры базы данных, даже если исходные таблицы разделяются, реструктуризуются или переименовываются. Заметим, однако, что определение представления должно быть обновлено, когда переименовываются лежащие в его основе таблицы или столбцы.

- **Целостность данных** Если доступ к данным или ввод данных осуществляется с помощью представления, СУБД может автоматически проверять, выполняются ли определенные условия целостности.

Недостатки представлений Наряду с перечисленными выше преимуществами, представления обладают и тремя существенными недостатками.

- **Производительность** Представление создает лишь видимость существования соответствующей таблицы, и СУБД приходится преобразовывать запрос к представлению в запрос к исходным таблицам. Если представление отображает многотабличный запрос, то простой запрос к представлению

становится сложным объединением и на его выполнение может потребоваться много времени. Однако это связано не с тем, что запрос обращается

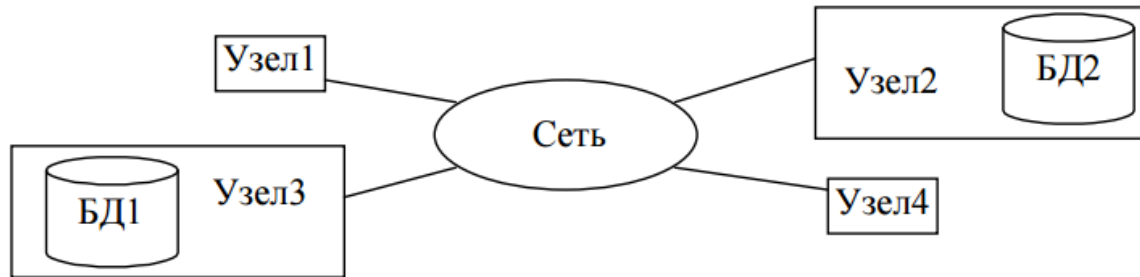
к представлению, — любой плохо построенный вопрос может вызвать проблемы с производительностью. Дело в том, что сложность запроса скрывается в представлении, так что пользователи не представляют, какой объем работы может вызвать даже кажущийся простым запрос.

- **Управляемость** Представления, как и все прочие объекты баз данных, должны быть управляемы. Если разработчики и пользователи баз данных смогут бесконтрольно создавать представления, то работа администратора базы данных станет существенно сложнее. Это в особенности справедливо в том случае, когда создаются представления, в основе которых лежат другие представления, которые, в свою очередь, могут быть основаны на других представлениях. Чем больше уровней между базовыми таблицами и представлениями, тем сложнее решать проблемы с представлениями, которые могут возникнуть в такой системе.

- **Ограничения на обновление** Когда пользователь пытается обновить строки представления, СУБД должна преобразовать запрос в запрос на обновление строк исходных таблиц. Это возможно для простых представлений; более сложные представления обновлять нельзя, они доступны только для выборки.

6. Распределенные БД

Под распределенной базой данных (РБД) понимается набор логически связанных между собой разделяемых данных, которые физически распределены по разным узлам компьютерной сети. СУРБД – это программный комплекс (СУБД), предназначенный для управления РБД и позволяющий сделать распределенность прозрачной для конечного пользователя. Прозрачность РБД заключается в том, что с точки зрения конечного пользователя она должна вести себя точно также, как централизованная. Логически единая БД разделяется на фрагменты, каждый из которых хранится на одном компьютере, а все компьютеры соединены линиями связи. Каждый из этих фрагментов работает под управлением своей СУБД.



Критерии распределенности (по К. Дейту):

- Локальная автономность. Локальные данные принадлежат локальным узлам и управляются администраторами локальных БД. Все процессы на локальном узле контролируются только этим узлом.
- Отсутствие опоры на центральный узел. В системе не должно быть узла, без которого система не может функционировать, т.е. не должно быть центральных служб.
- Непрерывное функционирование. Удаление или добавление узла не должно требовать остановки системы в целом.
- Независимость от местоположения. Пользователь должен получать доступ к любым данным в системе, независимо от того, являются эти данные локальными или удалёнными.
- Независимость от фрагментации. Доступ к данным не должен зависеть от наличия или отсутствия фрагментации и от типа фрагментации.
- Независимость от репликации. Доступ к данным не должен зависеть от наличия или отсутствия реплик данных.
- Обработка распределенных запросов. Система должна автоматически определять методы выполнения соединения (объединения) данных.
- Обработка распределенных транзакций. Протокол обработки распределённой транзакции должен обеспечивать выполнение четырёх основных свойств транзакции: атомарность, согласованность, изолированность и продолжительность.
- Независимость от типа оборудования. СУРБД должна функционировать на оборудовании с различными вычислительными платформами.
- Независимость от операционной системы. СУРБД должна функционировать под управлением различных ОС.
- Независимость от сетевой архитектуры. СУРБД должна быть способной функционировать в сетях с различной архитектурой и типами носителя.
- Независимость от типа СУБД. СУРБД должна быть способной функционировать поверх различных локальных СУБД, возможно, с различными моделями данных (требование гетерогенности)

Преимущества и недостатки РБД

Преимущества	Недостатки
Отражение структуры организации	Повышение сложности
Разделяемость и локальная автономность	Увеличение стоимости
Повышение доступности данных	Проблемы защиты
Повышение надежности	Усложнение контроля за целостностью данных
Повышение производительности	Отсутствие стандартов
Экономические выгоды	Недостаток опыта
Модульность системы	Усложнение процедуры разработки базы данных

Функции СУРБД

Основные функции – те же, что и у СУБД:

- Управление данными во внешней памяти
- Управлением буфером оперативной памяти
- Управление транзакциями
- Журнализация
- Поддержка языков языков БД

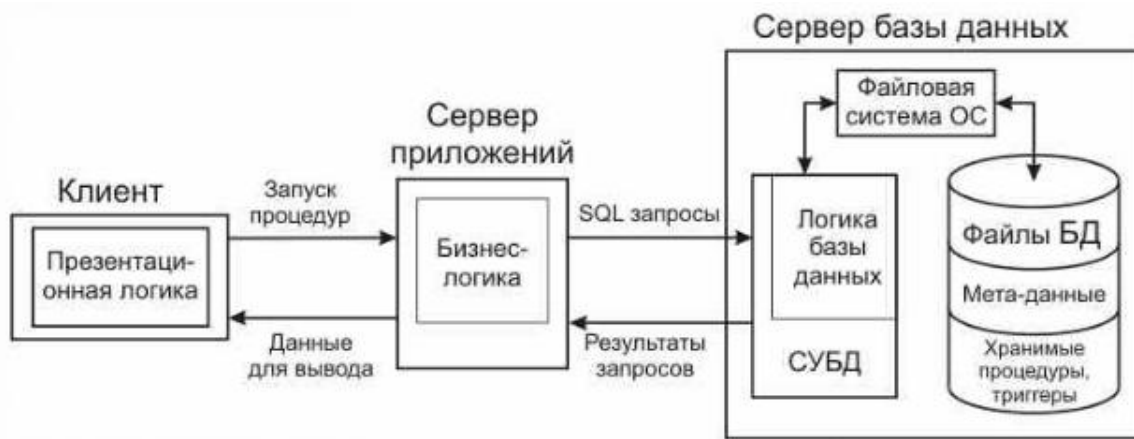
Дополнительные функции:

- 1)Расширенные службы установки соединений для передачи данных между узлами сети.
- 2)Расширенные средства ведения каталога, позволяющие сохранить сведения о распределении объектов БД в сети.
- 3)Средства обработки распределенных запросов, включающие механизмы оптимизации и организации удаленного доступа.
- 4)Функции, поддерживающие целостность реплицируемых данных.
- 5)Расширенные функции восстановления, учитывающие возможности отказов в других узлах сети или линиях связи.

Методы поддержки распределенных данных Существуют различные методы поддержки распределенности: 1. Фрагментация – разбиение БД или таблицы на несколько частей и хранение этих частей на разных узлах РБД. 2. Репликация – создание и хранение копий одних и тех же данных на разных узлах РБД. 3. Распределенные ограничения целостности – ограничения, для проверки выполнения которых требуется обращение к другому узлу РБД. 4. Распределенные запросы – это запросы на чтение, обращающиеся более чем к одному узлу РБД. 5. Распределенные транзакции – команды на изменение данных, обращающиеся более чем к одному узлу РБД.

7. Трехзвенная модель сервера приложений

Трёхуровневая архитектура (трёхзвенная архитектура) — архитектурная модель программного комплекса, предполагающая наличие в нём трёх компонентов: клиента, сервера приложений (к которому подключено клиентское приложение) и сервера баз данных (с которым работает сервер приложений).



Компоненты

1) Клиент (слой клиента) — это интерфейсный компонент комплекса, предоставляемый конечному пользователю. Этот уровень не должен иметь прямых связей с базой данных (по требованиям безопасности и масштабируемости), быть нагруженным основной бизнес-логикой и хранить состояние приложения (по требованиям надёжности). На этот уровень обычно выносятся только простейшая бизнес-логика: интерфейс авторизации, алгоритмы шифрования, проверка вводимых значений на допустимость и соответствие формату, несложные операции с данными (сортировка, группировка, подсчёт значений), уже загруженными на терминал.

2) Сервер приложений (средний слой, связующий слой) располагается на втором уровне, на нём сосредоточена большая часть бизнес-логики. Вне его остаются только фрагменты, экспортируемые на клиента (терминалы), а также элементы логики, погруженные в базу данных (хранимые процедуры и триггеры). Реализация данного компонента обеспечивается связующим программным обеспечением. Серверы приложений проектируются таким образом, чтобы добавление к ним дополнительных экземпляров обеспечивало горизонтальное масштабирование производительности программного комплекса и не требовало внесения изменений в программный код приложения.

3) Сервер баз данных (слой данных) обеспечивает хранение данных и выносятся на отдельный уровень, реализуется, как правило, средствами систем управления базами данных, подключение к этому компоненту обеспечивается только с уровня сервера приложений.

NB! Масштабируемость — способность системы справляться с увеличением рабочей нагрузки увеличивая производительность аппаратным или программным путём

8. Доступ данных в модели “Клиент-Интернет”(“Тонкий и толстый клиент”)

Тонкий клиент (англ. thin client) в компьютерных технологиях — компьютер или программа-клиент в сетях с клиент-серверной или терминальной архитектурой, который переносит все или большую часть задач по обработке информации на сервер. Данным термином может также называться P2P-клиент, использующий в качестве сервера другие узлы сети. **NB!** Клиент-сервер: нагрузка распределена между сервером и клиентом; все клиенты сети подключаются непосредственно к серверу (топология “звезда”) **NB!** Терминальная архитектура: все пользовательские данные и приложения размещаются на сервере; доступ к данным осуществляется посредством машин-терминалов. Под термином «тонкий клиент» подразумевается достаточно широкий с точки зрения системной архитектуры ряд устройств и программ, которые объединяются общим свойством: возможность работы в терминальном режиме. Таким образом, для работы тонкого клиента необходим терминальный сервер. Этим тонкий клиент отличается от толстого клиента, который, напротив, производит обработку информации независимо от сервера, используя последний в основном лишь для хранения данных. Тонкий клиент в большинстве случаев обладает минимальной аппаратной конфигурацией. **Толстый клиент** в архитектуре клиент-сервер — это приложение, обеспечивающее (в противовес тонкому клиенту) расширенную функциональность независимо от центрального сервера. Часто сервер в этом случае является лишь хранилищем данных, а вся работа по обработке и представлению этих данных переносится на машину клиента.

Достоинства:

- 1) Толстый клиент обладает широким функционалом в отличие от тонкого.
- 2) Режим многопользовательской работы.
- 3) Предоставляет возможность работы даже при обрывах связи с сервером.
- 4) Высокое быстродействие (зависит от аппаратных средств клиента)

Недостатки:

- 1) Большой размер дистрибутива.
- 2) Много в работе клиента зависит от того, для какой платформы он разрабатывался.
- 3) При работе с ним возникают проблемы с удаленным доступом к данным.
- 4) Довольно сложный процесс установки и настройки.
- 5) Сложность обновления и связанная с ней неактуальность данных.
- 6) Наличие бизнес-логики

9. Назначение интерфейса ODBC и его состав

ODBC (Open DataBase Connectivity) — это открытый интерфейс доступа к базам данных, разработанный фирмой X/Open. В начале 1990 г. существовало несколько поставщиков баз данных, каждый из которых имел собственный интерфейс. Если приложению было необходимо общаться с несколькими источниками данных, для взаимодействия с каждой из баз данных был необходим нестандартный код. Для решения возникшей проблемы Microsoft и ряд других компаний создали стандартный интерфейс для получения и отправки данных источникам данных различных типов. Этот интерфейс был назван open database connectivity (открытая связь с базами данных). С помощью ODBC программисты могли разрабатывать приложения для использования одного интерфейса доступа к данным, не беспокоясь о тонкостях взаимодействия с несколькими источниками. ODBC является интерфейсом уровня вызова, который позволяет приложениям иметь доступ к любой базе данных, для которой в ODBC есть драйвер. С помощью ODBC можно создать приложения для баз данных с доступом в любую базу данных, для которой у пользователя есть драйвер ODBC. ODBC обеспечивает API, который позволяет приложению быть независимым от системы управления базами данных (СУБД). ODBC представляет базы данных и является составной частью открытой архитектуры служб Microsoft Windows (WOSA), которая является интерфейсом, позволяющим приложениям Windows рабочего стола подключиться к IT-среде без переписывания приложения для каждой платформы.

Компоненты ODBC:

- **ODBC API**

Библиотека вызовов функций, набора кодов ошибок и стандартный синтаксис SQL-код для доступа к данным в СУБД.

- **Диспетчер драйверов ODBC**

Библиотека динамической компоновки (ODBC32.DLL), которая загружает драйвера базы данных ODBC от имени приложения. Библиотека DLL — прозрачна для приложения.

- **Драйвера базы данных ODBC**

Одна или несколько библиотек DLL, которые обрабатывают вызовы ODBC для конкретных СУБД. Список прилагаемых драйверов см. в разделе Список драйверов ODBC.

- **Библиотека курсоров ODBC**

Библиотека динамической компоновки (ODBCCR32.DLL), которая размещается между диспетчером драйверов и драйверами, обеспечивает прокрутку данных.

- **Администратор ODBC**

Инструмент используется для конфигурации СУБД, делает ее доступной для приложения.

Приложение добивается независимости от СУБД, работая через драйвер ODBC, написанный для конкретной СУБД, вместо работы непосредственно с СУБД. Драйвер преобразовывает вызовы в команды, пригодные к использованию в СУБД, упрощая работу разработчика, делая СУБД доступной

для значительного количества источников данных. Классы баз данных поддерживают все источники данных, для которых в ODBC есть драйвер. Источниками данных, например, могут быть: реляционная база данных, база данных индексно-последовательного метода доступа (ISAM), электронная таблица Microsoft Excel или текстовый файл. Драйверы ODBC управляют подключениями к источнику данных, для выбора записей из базы данных используется SQL.

10. Нормальные формы. Их назначение

Нормальная форма — свойство отношения в реляционной модели данных, характеризующее его с точки зрения избыточности, потенциально приводящей к логически ошибочным результатам выборки или изменения данных. Нормальная форма определяется как совокупность требований, которым должно удовлетворять отношение.

Процесс преобразования отношений базы данных к виду, отвечающему нормальным формам, называется нормализацией. Нормализация предназначена для приведения структуры БД к виду, обеспечивающему минимальную логическую избыточность, и не имеет целью уменьшение или увеличение производительности работы или же уменьшение или увеличение физического объёма базы данных.

Устранение избыточности производится, как правило, за счёт декомпозиции отношений таким образом, чтобы в каждом отношении хранились только первичные факты (то есть факты, не выводимые из других хранимых фактов).

Первая нормальная форма. Переменная отношения находится в первой нормальной форме (1НФ) тогда и только тогда, когда в любом допустимом значении отношения каждый его кортеж содержит только одно значение для каждого из атрибутов.

Условия нахождения в 1НФ:

1. Нет упорядочивания строк сверху вниз (другими словами, порядок строк не несет в себе никакой информации).
2. Нет упорядочивания столбцов слева направо (другими словами, порядок столбцов не несет в себе никакой информации).
3. Нет повторяющихся строк.
4. Каждое пересечение строки и столбца содержит ровно одно значение из соответствующего домена (и больше ничего).
5. Все столбцы являются обычными.

Пример:

<u>Сотрудник</u>	<u>Номер телефона</u>
Иванов И. И.	283-56-82 390-57-34
Петров П. П.	708-62-34

Вторая нормальная форма. Переменная отношения находится во второй нормальной форме тогда и только тогда, когда она находится в первой нормальной форме и каждый неключевой атрибут неприводимо (функционально полно) зависит от её потенциального ключа. Неприводимость означает, что в составе потенциального ключа отсутствует меньшее подмножество атрибутов, от которого можно также вывести данную функциональную зависимость. Вторая нормальная форма по определению запрещает наличие неключевых атрибутов, которые вообще не зависят от потенциального ключа. Таким образом, вторая нормальная форма в том числе запрещает создавать отношения как несвязанные (хаотические, случайные) наборы атрибутов.

Пример:

Филиал компании	Должность	Зарплата	Наличие компьютера
Филиал в Томске	Уборщик	20000	Нет
Филиал в Москве	Программист	40000	Есть
Филиал в Томске	Программист	25000	Есть

Третья нормальная форма. Переменная отношения находится в третьей нормальной форме тогда и только тогда, когда она находится во второй нормальной форме, и отсутствуют транзитивные функциональные зависимости неключевых атрибутов от ключевых.

Условие нахождения в НФ3:

1. Переменная отношения находится в НФ2
2. Ни один неключевой атрибут данной переменной отношения не находится в транзитивной функциональной зависимости от потенциального ключа этой же переменной отношений.

Пример:

Сотрудник	Отдел	Телефон
Гришин	Бухгалтерия	11-22-33
Васильев	Бухгалтерия	11-22-33
Петров	Снабжение	44-55-66

R2		R3	
Отдел	Телефон	Сотрудник	Отдел
Бухгалтерия	11-22-33	Гришин	Бухгалтерия
Снабжение	44-55-66	Васильев	Бухгалтерия
		Петров	Снабжение

Комментарий:

Первая таблица находится в НФ2. При разбиении на 2 подтаблицы, получаем НФ3

Опционально!

Нормальная форма Бойса-Кодда. Переменная отношения находится в нормальной форме Бойса — Кодда (иначе — в усиленной третьей нормальной форме) тогда и только тогда, когда каждая её нетривиальная и неприводимая слева функциональная зависимость имеет в качестве своего детерминанта некоторый потенциальный ключ.

Четвертая нормальная форма. Переменная отношения находится в четвёртой нормальной форме, если она находится в нормальной форме Бойса — Кодда и не содержит нетривиальных многозначных зависимостей.

Пятая нормальная форма. Переменная отношения находится в пятой нормальной форме (иначе — в проекционно-соединительной нормальной форме) тогда и только тогда, когда каждая нетривиальная зависимость соединения в ней определяется потенциальным ключом (ключами) этого отношения.

Шестая нормальная форма. Переменная отношения находится в шестой нормальной форме тогда и только тогда, когда она удовлетворяет всем нетривиальным зависимостям соединения. Из определения следует, что переменная находится в 6НФ тогда и только тогда, когда она неприводима, то есть не может быть подвергнута дальнейшей декомпозиции без потерь. Каждая переменная отношения, которая находится в 6НФ, также находится и в 5НФ.

11. Концептуальная, логическая и физическая модели БД.

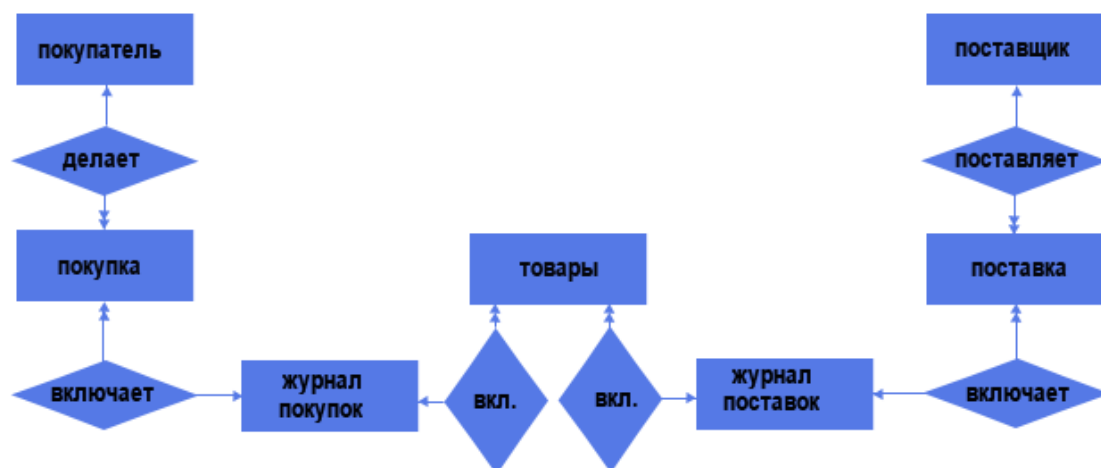
Начальной стадией проектирования системы баз данных является построение семантической модели предметной области, которая базируется на анализе свойств и природы объектов предметной области и информационных потребностей будущих пользователей разрабатываемой системы. Эту стадию принято называть концептуальным проектированием системы, а ее результат — концептуальной моделью предметной области (объектом моделирования здесь является предметная область будущей системы).

Моделирование предметной области базируется на использовании графических диаграмм, включающих сравнительно небольшое число компонентов, и самое важное – технологию построения таких диаграмм.

Чаще всего концептуальная модель базы данных включает в себя:

- описание информационных объектов или понятий предметной области и связей между ними.
- описание ограничений целостности, т.е. требований к допустимым значениям данных и к связям между ними.

(пример – Интернет-магазин). Так, все объекты, обозначающие вещи, обозначаются в виде прямоугольника. Атрибуты, характеризующие объект - в виде овала, а связи между объектами - ромбами. Мощность связи обозначаются стрелками (в направлении, где мощность равна многим - двойная стрелка, а со стороны, где она равна единице - одинарная).



Сущность, с помощью которой моделируется класс однотипных объектов, определяется как «предмет, который может быть четко идентифицирован». Также как каждый объект уникально характеризуется набором значений свойств, сущность должна определяться таким набором атрибутов, который позволял бы различать отдельные экземпляры сущностей. Каждый экземпляр сущности должен быть отличим от любого другого экземпляра той же сущности (это требование аналогично требованию отсутствия кортежей-дубликатов в реляционных таблицах). (к примеру – уникальный идентификатор таблицы).

Логическая модель данных. На следующем, более низком уровне находится логическая модель данных предметной области. Логическая модель описывает понятия предметной области, их взаимосвязь, а также ограничения на данные, налагаемые предметной областью. Примеры понятий - "сотрудник", "отдел", "проект", "зарплата". Примеры взаимосвязей между понятиями - "сотрудник числится ровно в одном отделе", "сотрудник может выполнять несколько проектов", "над одним проектом может работать несколько сотрудников". Примеры ограничений - "возраст сотрудника не менее 16 и не более 60 лет".

Логическая модель данных является начальным прототипом будущей базы данных. Логическая модель строится в терминах информационных единиц, но без привязки к конкретной СУБД. Более того, логическая модель данных необязательно должна быть выражена средствами именно реляционной модели данных. Основным средством разработки логической модели данных в настоящий момент являются различные варианты **ER-диаграмм (диаграммы сущность-связь)**. Одну и ту же ER-модель можно преобразовать как в реляционную модель данных, так и в модель данных для иерархических и сетевых СУБД, или в постреляционную модель данных.

Решения, принятые на предыдущем уровне, при разработке модели предметной области, определяют некоторые границы, в пределах которых можно развивать логическую модель данных, в пределах же этих границ можно принимать различные решения. Например, модель предметной области складского учета содержит понятия "склад", "накладная", "товар". При разработке соответствующей реляционной

модели эти термины обязательно должны быть использованы, но различных способов реализации тут много - можно создать одно отношение, в котором будут присутствовать в качестве атрибутов "склад", "накладная", "товар", а можно создать три отдельных отношения, по одному на каждое понятие.

Физическая модель данных. На еще более низком уровне находится физическая модель данных. Физическая модель данных описывает данные средствами конкретной СУБД. Мы будем считать, что физическая модель данных реализована средствами именно реляционной СУБД, хотя, как уже сказано выше, это необязательно. Отношения, разработанные на стадии формирования логической модели данных, преобразуются в таблицы, атрибуты становятся столбцами таблиц, для ключевых атрибутов создаются уникальные индексы, домены преобразуются в типы данных, принятые в конкретной СУБД.

Ограничения, имеющиеся в логической модели данных, реализуются различными средствами СУБД, например, при помощи индексов, декларативных ограничений целостности, триггеров, хранимых процедур. При этом опять-таки решения, принятые на уровне логического моделирования определяют некоторые границы, в пределах которых можно развивать физическую модель данных. Точно также, в пределах этих границ можно принимать различные решения. Например, отношения, содержащиеся в логической модели данных, должны быть преобразованы в таблицы, но для каждой таблицы можно дополнительно объявить различные индексы, повышающие скорость обращения к данным. Многое тут зависит от конкретной СУБД.

12. Типы связей: "один к одному" и др.

Что такое связи между таблицами

Связи. Концептуальная модель отражает связи между объектами разных классов. Связь определяется как «ассоциация, объединяющая несколько сущностей». Эта ассоциация всегда может существовать между разными сущностями или между сущностью и ею же самой (рекурсивная связь). В реляционной базе данных связи позволяют избежать избыточности данных.

Как и сущность, связь является типовым понятием, т.е. все экземпляры связываемых сущностей подчиняются правилам связывания типов.

Сущности, объединяемые связями, называются участниками. Степень связи определяется количеством участников связи.

Если каждый экземпляр сущности участвует, по крайней мере, в одном экземпляре связи, то такое участие этой сущности называется полным (или обязательным); в противном случае – неполным (или необязательным).

В реляционной базе данных связи позволяют избежать избыточности данных. Связь позволяет моделировать отношения между объектами предметной области. Наименование связи должно быть уникально во всей модели.

Связи с обеспечением целостности данных позволяют следить за тем, чтобы данные в одной таблице соответствовали данным в другой

Виды связей между таблицами

Связь осуществляется путем сопоставления данных в ключевых столбцах; обычно это столбцы, имеющие в обеих таблицах одинаковые названия. В большинстве случаев сопоставляются первичный ключ одной таблицы, содержащий для каждой из строк уникальный идентификатор, и внешний ключ другой таблицы.

Существует три вида связей между таблицами. Вид создаваемой связи зависит от того, как заданы связанные столбцы.

Связи "один ко многим"

Связь "один ко многим" - наиболее распространенный вид связи. При такой связи каждой строке таблицы А может соответствовать множество строк таблицы Б, однако каждой строке таблицы Б может соответствовать только одна строка таблицы А. Например, между таблицами "Издатели" и "Книги"

установлена связь "один ко многим": каждый из издателей может опубликовать множество книг, однако каждая книга публикуется лишь одним издателем.

Связь "один ко многим" создается в том случае, когда только на один из связываемых столбцов наложено ограничение уникальности или он является первичным ключом.

Связи "многие ко многим"

При установлении связи "многие ко многим" каждой строке таблицы А может соответствовать множество строк таблицы Б и наоборот. Такая связь создается при помощи третьей таблицы, называемой соединительной, первичный ключ которой состоит из внешних ключей, связанных с таблицами А и Б. Например, между таблицами "Авторы" и "Книги" установлена связь вида "многие ко многим", задаваемая с помощью связей вида "один ко многим" между каждой из этих таблиц и таблицей "АвторыКниг". Первичный ключ таблицы "АвторыКниг" - это сочетание столбцов "ИД_автора" (первичного ключа таблицы авторов) и "ИД_книги" (первичного ключа таблицы заголовков).

Связи "один к одному"

При установлении связи "один к одному" каждой строке таблицы А может соответствовать только одна строка таблицы Б и наоборот. Связь "один к одному" создается в том случае, когда оба связанные столбца являются первичными ключами или на них наложены ограничения уникальности.

Этот вид связи используется редко, поскольку в такой ситуации связываемые данные обычно можно хранить в одной таблице. Использовать связь вида "один к одному" можно в указанных ниже случаях.

- Чтобы разделить таблицу, содержащую слишком много столбцов.
- Чтобы изолировать часть таблицы по соображениям безопасности.
- Для хранения данных кратковременного использования, удалить которые проще всего путем очистки таблицы.
- Для хранения данных, имеющих отношение только к подмножеству основной таблицы.

13. Ссылочная целостность. Аномалии.

Ссылочная целостность

Ссылочной целостностью называют особый механизм, осуществляемый средствами СУБД или программистом, ответственный за поддержание непротиворечивых данных в связанных релятивных отношениях таблиц. **Ссылочная целостность** подразумевает, что в таблицах, имеющих релятивные связи, нет ссылок на несуществующие записи. Взгляните на рис. 1.3. Если мы удалим из списка студента Иванова И.И., и при этом не изменим таблицу со сданными экзаменами, ссылочная целостность будет нарушена, в таблице с экзаменами появится "мусор" - данные, на которые не ссылается ни одна запись из таблицы студентов. Ссылочная целостность будет нарушена.



Таким образом, если мы удаляем из списка студента Иванова И.И., следует позаботиться о том, чтобы из таблицы со сданными экзаменами также были удалены все записи, на которые ранее ссылалась удаленная запись главной таблицы. Существует несколько видов изменений данных, которые могут привести к нарушению ссылочной целостности:

1. Удаляется запись в родительской таблице, но не удаляются соответствующие связанные записи в дочерней таблице.

2. Изменяется запись в родительской таблице, но не изменяются соответствующие ключи в дочерней таблице.

3. Изменяется ключ в дочерней таблице, но не изменяется значение связанного поля родительской таблицы.

Многие СУБД блокируют действия пользователя, которые могут привести к нарушению связей. Нарушение хотя бы одной такой связи делает информацию в БД недостоверной. Если мы, например, удалили Иванова И.И., то теперь номер 1 принадлежит Петрову П.П.. Имеющиеся связи указывают, что он сдал экзамены по математике и физике, но не сдавал экзаменов по русскому языку и литературе. Достоверность данных нарушена. Конечно, в таких случаях в качестве ключа обычно используют счетчик - полеавтоинкрементного типа. Если удалить запись со значением 1, то другие записи не изменят своего значения, значение 1 просто невозможно будет присвоить какой-то другой записи, оно будет отсутствовать в таблице. Путаницы в связях не случится, однако все равно подчиненная таблица будет иметь "потерянные" записи, не связанные ни с какой записью главной таблицы. Механизм ссылочной целостности должен запрещать удаление записи в главной таблице до того, как будут удалены все связанные с ней записи в дочерней таблице.

Аномалии

-Аномалия обновления – появление в базе данных несогласованности данных при выполнении операций вставки, удаления, модификации записей.

-Аномалии модификации – появление записей с противоречащими значениями в некоторых столбцах при изменении значений соответствующих полей одной записи.

Для отношения Студент (ФИО, Группа, Староста), где в столбце Группа хранится полное название группы, а столбец Староста содержит ФИО старосты группы, изменение значения Староста (например, для устранения ошибки) может привести к существованию более одного старосты одной и той же группы.

-Аномалии удаления – удаление лишней информации при удалении записи.

Для отношения Студент (ФИО, Группа, Староста), удаление студента может привести к удалению из БД и ФИО старосты группы (в том случае, если для данной группы запись – единственная).

-Аномалии вставки – добавление лишней информации или возникновение противоречащих значений в некоторых столбцах при вставке новой записи.

Для отношения Студент (ФИО, Группа, Староста), где в столбце Группа хранится полное название группы, а столбец Староста содержит ФИО старосты группы, добавление названия новой группы повлечет обязательное определение ФИО студента и старосты, в то время как эти данные могут быть пока не известны. В то же время, при добавлении нового студента значение поля Староста в новой записи может не совпадать со значением данного поля для другого студента этой же группы.

Для сохранения корректности БД необходимо устранять данные аномалии, выполняя дополнительные операции по просмотру и модификации данных. Потери в производительности, вызванные выполнением действий по устранению аномалий, могут быть весьма существенными, при этом данные потери, в большинстве случаев, не являются неизбежными, а определяются неудачным выбором схемы БД.

Указанные аномалии связаны с избыточностью данных в БД. Следует различать избыточное и неизбыточное дублирование данных.

Неизбыточное дублирование возникает из необходимости хранить идентичные данные, поскольку важен сам факт их идентичности, и удаление хотя бы одного представителя идентичных данных приведет к невозможной потере информации. В качестве примера неизбыточного дублирования можно рассмотреть дублирование значений атрибута Группа в отношении Студент (ФИО, Группа). При удалении одного из повторяющихся значений Группы будет потеряна информация о принадлежности одного студента к данной группе.

Избыточное дублирование (избыточность) обычно связано с необходимостью задания значения всех атрибутов отношения, при этом дублируемые данные не являются необходимыми, и в случае потери (удаления) могут быть восстановлены по данным одного или нескольких отношений БД. в качестве примера избыточного дублирования можно рассмотреть указать дублирование значений атрибута Староста в отношении Студент (ФИО, Группа, Староста). Для каждого студента некоторой группы значение атрибута Староста может быть восстановлено по значениям данного атрибута других студентов рассматриваемой группы.

14. Конструктор форм

Окно конструктора форм

Любая форма в Visual FoxPro состоит из объектов, каждый из которых имеет характерные свойства. Для любого объекта вы можете указать действия, выполняемые программой при наступлении определенных событий. Процесс создания формы в конструкторе форм состоит в размещении в форме объектов и определении свойств, а также связанных с ними событий и выполняемых действий.

Для открытия окна конструктора форм при создании новой формы воспользуйтесь одним из следующих способов.

- Выполните команду New (Новый) из меню File (Файл). В открывшемся диалоговом окне New (Новый) выберите опцию Form (Форма) и нажмите кнопку New File (Новый файл).
- Нажмите кнопку на стандартной панели инструментов Visual FoxPro. В открывшемся диалоговом окне New (Новый) выберите опцию Form(Форма) и нажмите кнопку New File (Новый файл).
- Для размещения создаваемой формы в проекте выберите вкладку Documents (Документы), перейдите в группу Forms (Формы) и нажмите кнопку New окна проекта. Затем в открывшемся окне New Form (Новая форма) нажмите одноименную кнопку.

Компоненты:

Наименование	Назначение
Select Objects (Выбор объектов)	Указатель выделения. Позволяет выбирать в форме объекты
View Classes (Просмотр классов)	Позволяет выбрать класс для создаваемых в форме объектов
Label (Метка)	Создает в форме текстовый объект
Text Box (Поле ввода)	Создает в форме поле ввода
Edit Box (Поле редактирования)	Создает в форме поле редактирования
Command Button (Кнопка)	Создает в форме кнопку управления
Option Group (Переключатель)	Создает в форме переключатель
Check Box (Флажок)	Создает в форме флажок
Grid (Таблица)	Создает в форме для размещения полей объект в виде таблицы
Combo Box (Раскрывающийся список)	Создает в форме раскрывающийся список
List Box (Список)	Создает в форме список
Spinner (Счетчик)	Создает в форме поле ввода значения в виде счетчика
Line (Линия)	Создает в форме линию
Shape (Контур)	Создает в форме контур
Container (Контейнер)	Создает в форме контейнер
Image (Изображение)	Размещает в форме рисунок
Command Group (Группа кнопок)	Размещает в форме группу кнопок
Timer (Таймер)	Создает в форме объект типа таймера
Page Frame (Вкладка)	Размещает в форме страницы с вкладками
ActivX Bound Control(OleBoundControl) (ActivX-объект)	Отображает содержимое OLE-объекта, хранящегося в поле типа General
ActivX Control(OleControl) (OLE-объект)	Создает OLE-объект
HyperLink (Гиперссылка)	Создает объект для работы с Интернетом
Separator (Разделитель)	Размещает на панели инструментов разделитель кнопок
Builder Lock (Закрепитель строителя)	Закрепляет выбор строителя
Button Lock (Закрепитель кнопки)	Закрепляет выбранную кнопку на панели инструментов

Для выравнивания объектов, размещенных в форме, удобно использовать панель инструментов Layout (Расположение). Краткое назнач

Таблица 6.3. Кнопки панели инструментов Layout

Наименование	Назначение
Align Left Sides (По левому краю)	Выравнивает выбранные объекты по левому краю самого левого объекта
Align Top Edges (По верхнему краю)	Выравнивает выбранные объекты по верхнему краю самого верхнего объекта
Align Right Sides (По правому краю)	Выравнивает выбранные объекты по правому краю самого правого объекта
Align Bottom Edges (По нижнему краю)	Выравнивает выбранные объекты по нижнему краю самого нижнего объекта
Align Vertical Centers (По вертикали)	Выравнивает выбранные объекты по вертикали
Align Horizontal Centers (По горизонтали)	Выравнивает выбранные объекты по горизонтали
Center Vertically (По вертикальному центру)	Центрирует выбранные объекты относительно вертикальной средней линии формы
Center Horizontally (По горизонтальному центру)	Центрирует выбранные объекты относительно горизонтальной средней линии
Same Width (Одинаковая ширина)	Устанавливает одинаковую ширину для выбранных объектов формы
Same Size (Одинаковый размер)	Устанавливает одинаковую ширину и высоту для выбранных объектов формы
Same Height (Одинаковая высота)	Устанавливает одинаковую высоту для выбранных объектов формы
Send to Back (Позади)	Направляет выбранный объект на самый нижний слой формы
Bring to Front (Поверх)	Направляет выбранный объект на самый верхний слой формы

Процесс создания формы включает следующие действия:

- настройка параметров формы;
- определение среды окружения, т. е. выбор используемых в форме таблиц и установка связей между ними;
- размещение в форме объектов: текста, полей различных типов, линий, рисунков, кнопок управления;
- настройка свойств размещенных в форме объектов.

Форма, как и все располагаемые в ней объекты, имеет свойства, используя которые можно задать ее размер, координаты верхнего левого угла, стиль рамки обрамления, заголовок, цвет и т. д.

Настройка параметров формы осуществляется в окне свойств Properties(Свойства), для открытия которого установите курсор на свободную от объектов поверхность формы, нажмите правую кнопку мыши и выберите из контекстного меню команду Properties (Свойства).

(далее следует описать все свойства формы. Ноутбук с FoxPro в помощь).

15. Элементы управления(классы) в VFP.

Форма в чистом виде, без элементов управления и данных, которыми она должна манипулировать, интереса не представляет. Поэтому любая форма создаётся именно с целью оперирования данными с помощью элементов управления, включаемых в форму. Данные в VFP хранятся, естественно, в таблицах и подключаются к форме с помощью специального объекта Data environment

Наименование	Назначение	Видимый	Контейнер
ActiveDoc	Активный документ	Нет	Нет
CheckBox	Флажок	Да	Нет
Cplumn	Столбец	Да	Да
ComboBox	Раскрывающийся список	Да	Нет
CommandButton	Кнопка управления	Да	Нет
CommandGroup	Набор кнопок управления	Да	Да
Container	Контейнер	Да	Да
Control	Базовый визуальный класс	Да	Нет
Custom	Базовый невидимый класс	Нет	Нет
EditBox	Поле редактирования	Да	Нет
Form	Форма	Да	Да
FormSet	Набор форм	Нет	Да
Grid	Таблица	Да	Да
Header	Заголовок столбцов таблицы	Да	Нет

Наименование	Назначение	Видимый	Контейнер
HyperLink Object	Гиперссылка	Нет	Нет
Image	Изображение	Да	Нет
Label	Надпись	Да	Нет
Line	Линия	Да	Нет
ListBox	Список	Да	Нет
OleContainerControl	OLE-объект управления		
OleBoundControl	OLE-объект данных		
OptionButton	Переключатель	Да	Нет
OptionGroup	Набор переключателей	Да	Да
Page	Вкладка формы	Да	Да
PageFrame	Макет страницы	Нет	Да
ProjectHook	Проект	Нет	Да
Separator	Разделитель	Да	Нет
Shape	Обрамление	Да	Нет
Spinner	Счетчик	Да	Нет
TextBox	Поле ввода	Да	Нет
Timer	Таймер	Нет	Нет
ToolBar	Панель управления	Да	Да

16. Конструктор меню.

Создание меню

После того как вы открыли конструктор, можно приступить к созданию меню. Для этого выполните следующие действия:

1. В поле Prompt (Приглашение) введите наименования первого пункта меню и нажмите клавишу <Enter> или <Tab> для перехода на следующее поле. Курсор оказывается в списке Result (Результат).
2. Для определения типа пункта меню (табл. 11.2) нажмите кнопку раскрытия списка и выберите необходимое значение из тех, которые предлагает система.

Таблица 11.2. Типы пунктов меню

Тип меню	Назначение
Command (Команда)	При выборе пункта меню данного типа будет выполняться связанная

		с ним команда
Pad (Наименование строки меню)	Name	При выборе пункта меню никаких действий выполняться не будет. Как правило, используется в качестве дополнительного пояснения к меню
Submenu (Подменю)		При выборе пункта меню раскрывается связанное с данным пунктом ниспадающее меню
Procedure (Процедура)		При выборе пункта меню вызывается процедура, определенная для данного пункта меню

Замечание

При использовании значения типа Command (Команда) с правой стороны появляется поле для ввода команды, выполняемой при выборе данного пункта меню. Это может быть, например, команда вызова формы или формирования отчета. Если из списка Result (Результат) вы выбрали значение Procedure (Процедура) или Submenu (Подменю), в окне конструктора правее описания типа пункта меню появляется кнопка Create (Создать). При нажатии на эту кнопку вы переходите, соответственно, в окно создания процедуры или в окно создания ниспадающего меню для выбранного пункта меню.

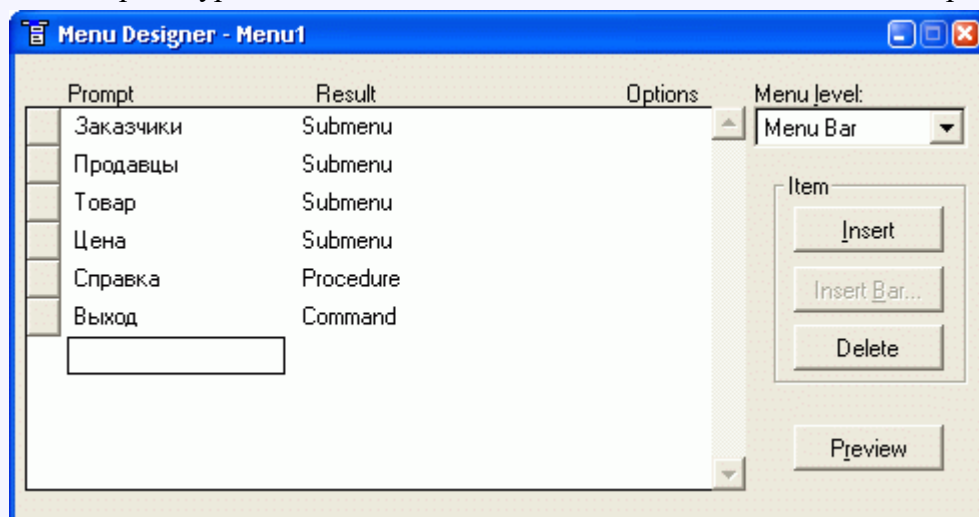


Рис. 11.3. Основное меню приложения

- Указав тип пункта меню, перейдите в следующую строку и введите информацию о втором пункте меню.
- Введите наименования остальных пунктов меню и их типы (рис. 11.3).
- Для просмотра созданных пунктов меню нажмите кнопку Preview (Просмотр). Основное меню Visual FoxPro будет заменено созданным меню. Пункты меню отображаются на экране в порядке их описания. На экране также появляется диалоговое окно Preview (Просмотр), в котором отображается текст текущего пункта меню, его тип и выполняемое действие (рис. 11.4).

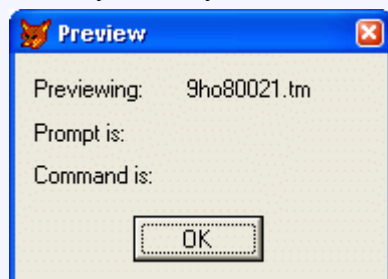


Рис. 11.4. Просмотр созданного меню приложения

17. Конструктор представлений

Создание представления данных

Для создания представления используется конструктор представлений данных. Существует несколько альтернативных способов его вызова.

- Выберите в меню File (Файл) команду New (Новый). В открывшемся диалоговом окне установите опцию View (Представление данных) и нажмите кнопку New file (Новый файл).
- В окне конструктора проекта на вкладке Data (Данные) перейдите в раздел Local Views (Локальные представления данных) выбранной базы данных и нажмите кнопку New (Новый).
- Введите в командном окне Command команду CREATE VIEW.

Замечание

При любом из вариантов вызова для создания представления данных вы можете воспользоваться услугами мастера, который аналогичен мастеру создания запросов.

При вызове конструктора представлений открывается диалоговое окно Add Table and View (Добавить таблицу или представление данных), позволяющее разместить в конструкторе таблицы и созданные ранее представления данных. Для отображения представлений в этом диалоговом окне необходимо в области Select (Выбор) установить опцию Views (Представления данных). Выбрав необходимую таблицу или представление данных, перенесите их в конструктор, нажав кнопку Add (Добавить). Сформировав список таблиц, нажмите кнопку Close (Заккрыть) для закрытия диалогового окна Add Table and View (Добавить таблицу или представление). Выбранные таблицы размещаются в открывшемся на экране окне конструктора представлений данных (рис. 14.7).

Окно конструктора представлений данных похоже на окно конструктора запросов и отличается от последнего некоторыми дополнительно предоставляемыми функциями.

- Можно указать поля для модификации данных.
- Можно задать параметры, значения которых будут запрашиваться при открытии представления данных.
- Можно определить количество выбираемых записей.

Для формирования представления данных вы можете использовать команды меню Query (Запрос) и кнопки на панели инструментов View Designer (Конструктор представления данных). Функции кнопок панели инструментов View Designer описаны в табл. 14.2.

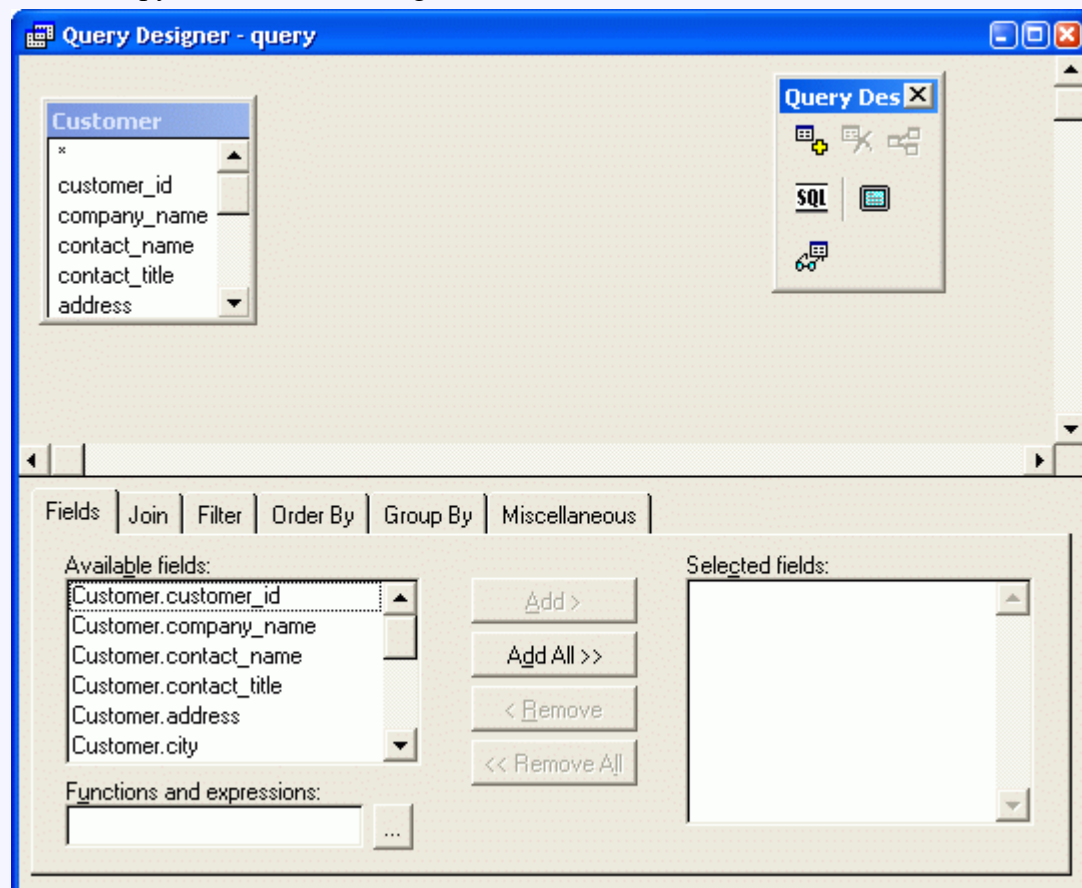


Рис. 14.7. Окно конструктора представлений данных

Таблица 14.2. Назначение кнопок панели инструментов View Designer

Название	Назначение
Add Table (Добавить таблицу)	Добавляет в представление новую таблицу или представление данных
Remove Table (Удалить таблицу)	Удаляет выбранную таблицу из запроса
Add Join (Добавить объединение)	Открывает диалоговое окно Join Condition (Условие объединения) для задания условия объединения таблиц
Show the SQL windows (Показать SQL-оператор)	Открывает диалоговое окно, в котором отображается SQL-оператор, соответствующий созданному представлению данных
Maximize the table view (Раскрыть панель представления таблиц)	Раскрывает на весь экран панель отображения используемых в представлении таблиц. Повторное нажатие на эту кнопку возвращает панели первоначальный размер

18. Конструктор отчетов

Создание отчета с помощью конструктора отчетов

В предыдущей главе мы рассмотрели создание отчета с помощью мастера. В процессе использования отчета практически всегда возникает необходимость его модификации. Создавать сложные отчеты с помощью мастера не удастся. И в этих случаях вам не обойтись без конструктора отчетов.

Окно конструктора отчетов

Существует несколько различных вариантов открытия окна конструктора отчетов. Если вы хотите модифицировать ранее созданный отчет, то в окне проекта установите курсор на его названии и нажмите кнопку Modify (Модификация).

Для открытия окна конструктора отчетов при создании нового отчета выполните одно из следующих действий.

- В меню File (Файл) выберите команду New (Новый). В открывшемся диалоговом окне New (Новый) выберите опцию Report (Отчет) и нажмите кнопку New file (Новый файл).
- Нажмите кнопку New (Новый) в окне проекта, предварительно выбрав группу Reports (Отчеты).
- Нажмите кнопку New (Новый) на стандартной панели инструментов; в открывшемся диалоговом окне New (Новый) установите опцию Report (Отчет) и нажмите кнопку New file (Новый файл).

Для работы в конструкторе отчетов используются панели инструментов Report Designer (Конструктор отчета) и Report Controls (Элементы управления отчета), а также команды пункта Report (Отчет) (рис. 8.1), появившегося в строке основного меню при открытии конструктора.

В табл. 8.1 приведено краткое описание кнопок панели инструментов Report Controls (Элементы управления отчета). Более подробно назначение кнопок этой панели будет рассмотрено в разделах, посвященных размещению в отчетах различных элементов управления.

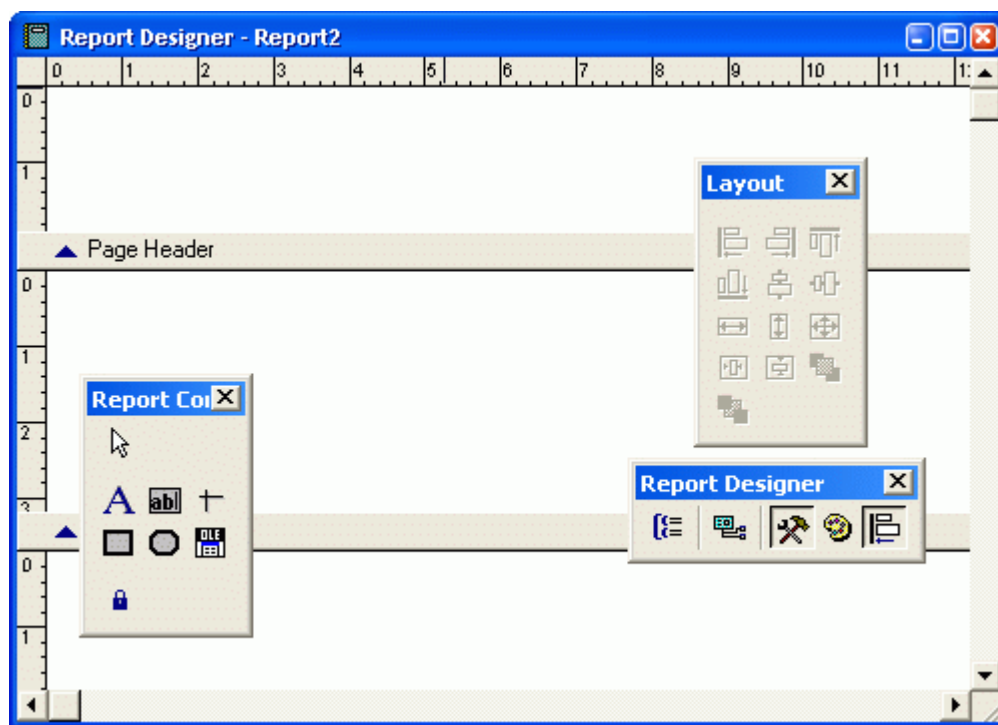


Рис. 8.1. Окно конструктора отчета

Таблица 8.1. Кнопки панели инструментов Report Controls

Наименование	Назначение
Select Objects (Выбор объектов)	Является указателем выбора объектов отчета
Label (Метка)	Размещает текст
Field (Поле)	Размещает поля
Line (Линия)	Рисует линии
Rectangle (Прямоугольник)	Рисует прямоугольники
Rounded Rectangle (Скругленный прямоугольник)	Рисует прямоугольник со скругленными краями
Picture/ActiveX Bound Control (Изображение/ActiveX-объект)	Помещает в отчет рисунок
Button Lock (Закрепитель кнопки)	Закрепляет выбор кнопки

19. Конструктор запросов.

Запросы к базе данных

Одним из основных назначений разработанного приложения является быстрый поиск информации в базе данных и получение ответов на разнообразные вопросы. Для этих целей в Visual FoxPro используются средства, называемые запросами.

Для решения таких задач предназначен конструктор запросов и команда SELECT языка Visual FoxPro.

С помощью конструктора запросов Visual FoxPro вы можете формировать различной сложности критерии для выбора записей из одной или нескольких таблиц, указывая при этом, какие поля должны

быть отображены в запросе. Над полями, выбираемыми из таблиц с помощью запросов, можно выполнять различные вычисления.

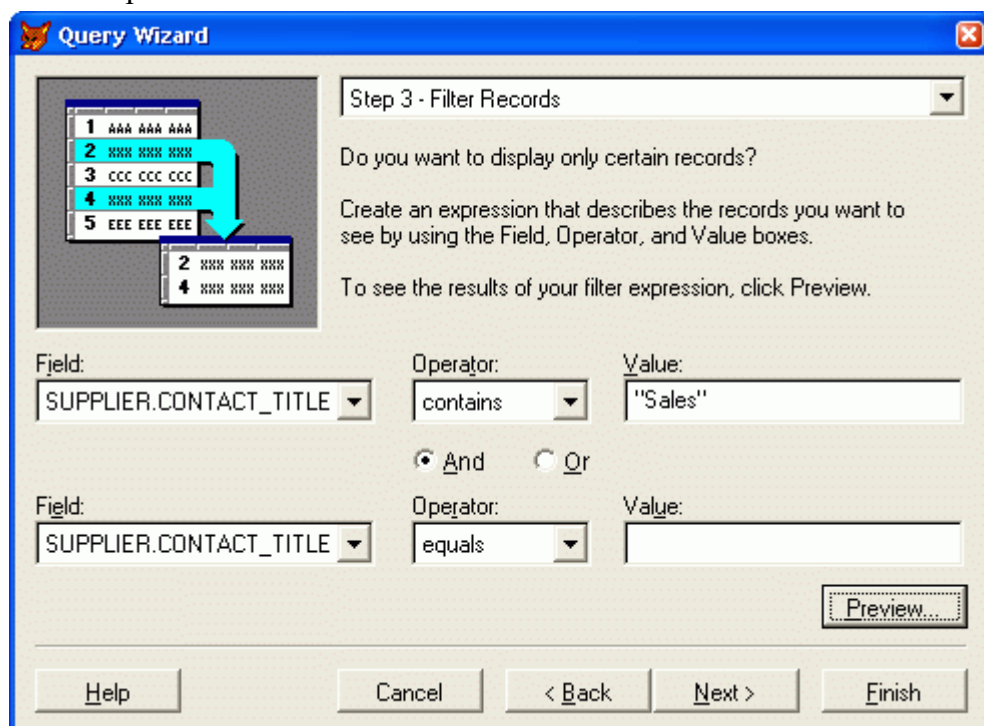


Рис. 9.1. Ввод условия выборки в мастере запросов

Результатом запроса является таблица, которую вы можете сохранить в массиве, в создаваемой новой таблице, отобразить на экране в режиме Browse (Просмотр) или вывести в виде отчета.

Для создания запросов вы можете использовать мастер запросов, который последовательно запрашивает наименования таблиц, используемых в запросе, перечень полей таблиц, критерий упорядочения и условия фильтрации данных. На рис. 9.1 приведено диалоговое окно мастера, позволяющее сформировать условия фильтрации выбираемых из таблицы данных. Мы не будем рассматривать создание запросов с помощью мастера, т. к. конструктор запросов достаточно прост и работа в нем у вас не вызовет затруднений.

Конструктор запросов

Для создания запроса в окне конструктора запросов выполните следующие действия:

1. На вкладке Data (Данные) конструктора проекта выберите группу Queries (Запросы).
2. Нажмите кнопку New (Новый).
3. В открывшемся диалоговом окне New Query (Новый запрос) нажмите кнопку New Query (Новый запрос). Открывается диалоговое окно выбора таблиц Add Table or View (Добавить таблицу или представление данных).
4. В этом диалоговом окне выберите таблицы, данные из которых хотите использовать в запросе, и с помощью кнопки Add (Добавить) перенесите их в окно конструктора запросов.
5. Завершив выбор таблиц, нажмите кнопку Close (Заккрыть).

На экране появляется окно конструктора запросов (рис. 9.2), которое содержит названия выбранных таблиц, а в основном меню появляется пункт Query (Запрос). Можно приступить к формированию условий запроса.

Совет

Для открытия ранее созданного запроса в окне конструктора запросов на вкладке Data (Данные) окна проекта в группе Queries (Запросы) найдите модифицируемый запрос, установите на него курсор и нажмите кнопку Modify (Модифицировать).

Далее, открывая в конструкторе запросов необходимые вкладки, вы выполняете следующие действия:

- выбираете поля результирующей таблицы запроса;

- формируете вычисляемые поля;
- указываете критерии для выборки, группировки и упорядочения данных;
- задаете, куда выводить результат выборки.

В верхней части окна конструктора запросов расположена панель, на которой отображаются используемые в запросе таблицы. Ниже находятся вкладки, предназначенные для выбора полей запроса и формирования условий выборки. Назначение этих вкладок приведено в табл. 9.1.

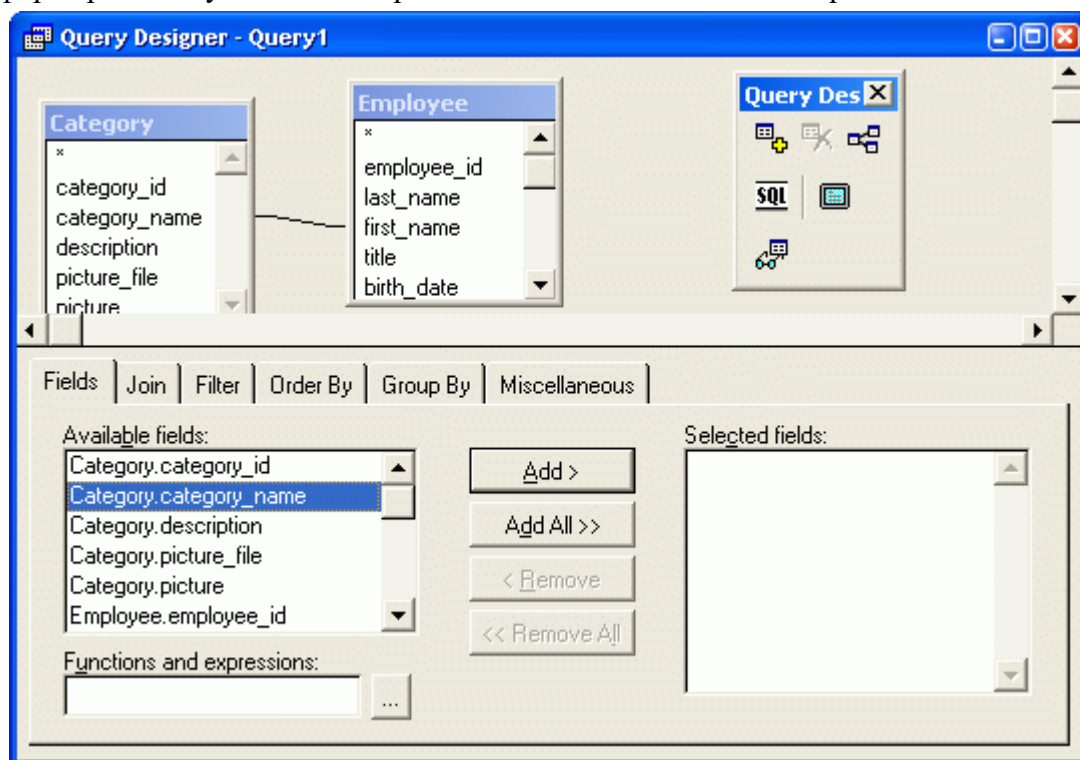


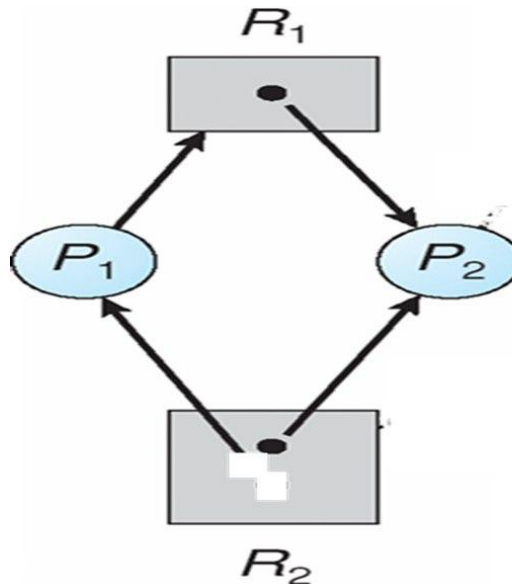
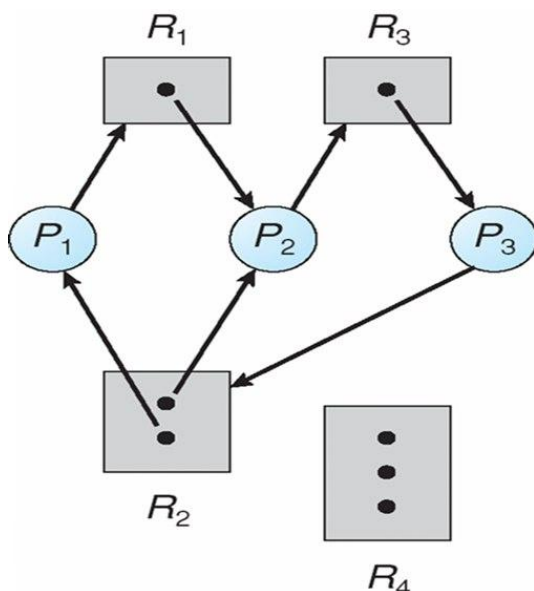
Рис. 9.2. Окно конструктора запросов с выбранной таблицей клиентов

Таблица 9.1. Назначение вкладок окна конструктора запросов

Вкладка	Назначение
Fields (Поля)	Позволяет указать поля исходных таблиц, выбираемые в результирующий запрос
Join (Объединение)	Позволяет задать условия объединения таблиц
Filter (Фильтр)	Позволяет определить фильтры, накладываемые для выбора записей
Order By (Упорядочение)	Позволяет задать критерии упорядочения данных
Group By (Группировка)	Позволяет задать условия группировки данных
Miscellaneous (Разное)	Позволяет задать дополнительные условия, такие как признак выборки повторяющихся значений, количество или процент выбора данных

Общая модель блокировок

- System consists of resources
- (Система состоит из ресурсов)
- Resource types R_1, R_2, \dots, R_m
CPU cycles, memory space, I/O devices
(Ресурсы имеют тип R_1, R_2, \dots, R_m
Захват процессора, области памяти, устройств ввода-вывода)
- Each resource type R_i has W_i instances.
- Любой ресурс типа R_i имеет W_i экземпляров.
- Each process utilizes a resource as follows:
- Каждый процесс использует ресурс следующим образом
 - **request** –запрос(требование)
 - **use** -использование
 - **Release**-освобождение



21. Триггеры и хранимые процедуры.

В Visual FoxPro для определения триггеров введите в поля ввода Insert trigger (Вставить триггер), Update trigger (Обновить триггер) или Delete trigger (Удалить триггер) вкладки Table (Таблица) окна конструктора выбранной таблицы операторы сравнения, вызов хранимой процедуры или любое логическое выражение. Если результат вычисления выражения равен True (Истина), то считается, что введены допустимые значения. В противном случае сохранения введенных данных не происходит и формируется сообщение об ошибке. При использовании хранимых процедур вы сможете не только проверить условие достоверности ввода данных, но и задать действия, выполняемые при добавлении, удалении и изменении данных.

Вызов триггера Delete (Удалить) осуществляется:

- при выполнении команды DELETE;
- когда вы помечаете запись на удаление в режиме Browse (Обзор) или Edit (Правка).

Вызов триггера insert (Вставить) осуществляется в следующих случаях:

- при выполнении команд APPEND FROM, APPEND FROM ARRAY, APPEND BLANK;

- при добавлении в таблицу новой записи в режиме Browse (Обзор) или Edit (Правка);
- при выполнении команд IMPORT, INSERT SQL, RECALL;
- при снятии метки об удалении записи в режиме Browse (Обзор) или Edit (Правка).

Триггер update (Обновить) вызывается, когда:

• наступает любое событие, которое приводит к модификации записи (например, при изменении значения поля);

- выполняются команды GATHER, REPLACE, REPLACE FROM ARRAY, UPDATE SQL.

При использовании триггеров необходимо учитывать ограничения, имеющиеся в Visual FoxPro.

• При модификации записей, помеченных для удаления, и выполнении команды PACK триггеры не вызываются.

- Выполнение команды ZAP не вызывает триггер Delete (Удалить).
- При использовании буферизации ввода триггер update (Обновить) вызывается только при вызове функции TABLEUPDATE (),

Для удаления триггера в окне конструктора перейдите на вкладку Table (Таблица) и очистите поле ввода выражения для триггера или используйте Команду DELETE TRIGGER.

Хранимые процедуры

Для создания хранимой процедуры выполните следующие действия:

1. В окне проекта выберите базу данных.
2. Перейдите в группу Stored Procedures (Хранимые процедуры).
3. Нажмите кнопку New (Новый)

Для создания хранимой процедуры предназначена группа Stored Procedures окна проекта

Открывается окно редактирования хранимых процедур (рис. 14.3), которое содержит все ранее созданные хранимые процедуры, включая процедуры, создаваемые автоматически при определении условий целостности базы данных.

Предупреждение

Редактирование или удаление хранимых процедур, которые Visual FoxPro создал автоматически при определении условия целостности данных, могут привести к непредсказуемым последствиям.

Для редактирования хранимых процедур в Visual FoxPro вы можете использовать команду MODIFY PROCEDURE, которая открывает окно редактирования хранимых процедур текущей базы данных.

Для удаления хранимой процедуры необходимо в окне редактирования выделить текст удаляемой процедуры и нажать клавишу <Delete> или в окне конструктора проекта установить курсор на ее наименование и выполнить команду Remove (Удалить) окна проекта.

22. OLE-объекты.

В Visual FoxPro OLE-объекты могут храниться и отображаться только в полях типа General (Общий). Для внедрения OLE-объекта в таблицу Visual FoxPro выполните следующие действия:

1. Создайте в таблице, предназначенной для хранения OLE-объекта, поле типа General.
2. Откройте таблицу в режиме Browse (Обзор) (рис. 21.1), выполнив команду Browse (Обзор) из меню View (Вид) или нажав кнопку Browse (Обзор) окна проекта. Откроется окно просмотра таблицы.

Замечание

Если поле типа General (Общий) пусто, то при просмотре таблицы в нем появляется пометка gen. Если поле содержит OLE-объект, то пометка принимает вид Gen.

3. Для вставки OLE-объекта дважды щелкните на поле типа General (Общий), содержащем значение gen. Откроется окно редактирования этого поля (рис. 21.2).

4. В меню Edit (Правка) выберите команду Insert Object (Вставить объект). Откроется диалоговое окно Вставка объекта (рис. 21.3).

5. Это диалоговое окно содержит два переключателя, предоставляющих два варианта включения OLE-объекта:

- Создать новый — позволяет создать новый OLE-объект и вставить его в поле;
- Создать из файла — позволяет разместить в поле таблицы OLE-объект из файла.

При выборе переключателя Создать новый выберите из предложенного списка Тип объекта сервер, который хотите использовать для создания объекта, и нажмите кнопку ОК. Откроется приложение, в котором создайте объект, после чего вернитесь в Visual FoxPro.

Для использования существующего объекта выберите переключатель Создать из файла. Диалоговое окно примет другой вид. Воспользовавшись кнопкой Обзор, найдите необходимый файл и вставьте его содержимое в поле таблицы.

Замечание

В Visual FoxPro для хранения сложных текстов можно использовать поле типа General (Общий), внедрив в него или связав с ним документ текстового процессора.

23. Ограничение прав доступа. Фильтры.

Какие преимущества использования ограничений доступа?

Во-первых, уменьшаются затраты по эксплуатации системы в целом, поскольку не возникает возможность некорректного ввода или редактирования пользователем с недостаточной компетенцией документов, к которым он отношения не имеет.

Во-вторых, когда есть необходимость ограничить доступ к просмотру информации пользователя по определенным критериям. К примеру, при филиальной структуре предприятия не обязательно кассиру филиала 1 видеть остатки по кассе филиала 2.

В-третьих, использование механизма установки границы запрета редактирования данных предупреждает "случайное" изменение данных прошлых периодов.



Однопользовательская база данных создается для монопольного использования одним приложением, функционирующим, как правило, на том же компьютере, на котором хранится сама база данных.

Многопользовательская база данных рассчитана на несколько приложений, которые могут работать с разными подсхемами этой базы данных.

Многопользовательский режим может допускать параллельный или последовательный доступ. При последовательном доступе база данных «захватывается» очередным приложением на все время его работы.

При параллельном доступе база данных становится разделяемым ресурсом, к которому одновременно могут обращаться разные приложения.

База данных, хранящаяся на одном компьютере, называется централизованной. В более сложном случае многопользовательская база данных с параллельным доступом может быть разделена на фрагменты, которые хранятся отдельно на разных компьютерах. Такие базы данных называются распределенными.

Ограничение (constraint) — это условие, которому должны удовлетворять возможные значения, хранящиеся в поле.

Существуют следующие типы ограничений

- ограничение по сущностям
- ограничение по ссылкам
- пользовательские ограничения

Ограничения по сущностям и по ссылкам мы уже рассмотрели. Пользовательские ограничения позволяют пользователю наложить условия на значения данных, хранящихся в поле.

Существуют базы данных, которые не поддерживают ограничений (к таким СУБД, в частности, относятся dBase III, Clipper и FoxPro). В этом случае необходимо помещать код, реализующий ограничения в клиентские приложения, либо применять другие объекты баз данных, например, триггеры, для того, чтобы обеспечить те же функции.

Права на доступ к объектам баз данных.

Для различных объектов применяются разные наборы прав доступа к ним:

- SELECT, INSERT, UPDATE, DELETE, REFERENCES- эти права могут быть применены для таблицы или представления;

- SELECT и UPDATE — эти права могут быть применены к конкретному столбцу таблицы или представления;

- INSERT и DELETE — эти права применяются для таблицы или представления;

- EXECUTE — это право применяется только к конкретным хранимым процедурам и функциям, разрешая пользователю их выполнение.

- REFERENCES- возможность ссылаться на указанный объект. Применительно к таблицам разрешает пользователю создавать внешние ключи, ссылающиеся на первичный ключ или уникальный столбец этой таблицы. Применительно к представлениям право REFERENCES позволяет связывать представление со схемами таблиц, на основе которых строится представление. Это позволяет отслеживать изменения структуры исходных таблиц, которые могут повлиять на работу представления. Право REFERENCES не существовало в предыдущих версиях SQL Server.

Доступ можно предоставлять как на уровне всей таблицы или представления, так и на уровне отдельного столбца. Если в таблице имеется один или более столбцов, доступ пользователей к которым необходимо ограничить, то в базе данных создается представление, включающее только те столбцы исходной таблицы, доступ к которым необходимо разрешить.

Для управления разрешениями пользователя на доступ к объектам базы данных используется команда GRANT.

В дополнение к рассмотренному в команде указывается имя того объекта системы безопасности, который необходимо включить в роль. В качестве таких объектов могут выступать как учетные записи SQLServer, так и пользователи и группы пользователей, которым предоставлен доступ к серверу баз данных.

Права на исполнение команд Transact-SQL. Этот класс прав контролирует возможность создания объектов в базе данных, создания самой базы данных и выполнения процедуры резервного копирования.

Неявные права. Выполнение некоторых действий не требует явного разрешения и доступно по умолчанию. Третий класс прав как раз предоставляет такие действия. Эти действия могут быть выполнены только членами ролей сервера или владельцами объектов в базе данных.

Неявные права не предоставляются пользователю непосредственно, он получает их лишь при определенных обстоятельствах.

Для конкретного действия, контролируемого разрешением, возможны три варианта состояния доступа: предоставление, запрещение и неявное отклонение.

Для запрещения пользователю доступа к объектам базы данных используется команда DENY.

Параметры данной команды аналогичны параметрам команды GRANT. Использование параметра CASCADE позволяет отзывать права не только у данного пользователя, но также и у всех тех пользователей, кому он предоставил данные права.

Для неявного отклонения доступа к объектам базы данных можно использовать команду REVOKE, синтаксис которой:

```
REVOKE [GRANT OPTION FOR]
{ALL | PRIVILEGES}
```

Параметры имеют смысл, аналогичный параметрам команд GRANT и DENY. Параметр GRANT OPTION FOR используется, когда необходимо отозвать право, предоставленное параметром WITH GRANT OPTION команды GRANT. Пользователь при этом сохраняет разрешение на доступ к объекту, но теряет возможность предоставлять это разрешение другим пользователям.

Неявное отклонение доступа позволяет более гибко конфигурировать систему безопасности.

Конфликты доступа. Разрешения, предоставленные роли или группе, наследуются их членами. Хотя пользователю может быть предоставлен доступ через членство в одной роли, роль другого уровня может иметь запрещение на действия с объектом. В таком случае возникает конфликт доступа.

При разрешении конфликтов доступа SQL Server руководствуется принципом, состоящим в том, что разрешение на предоставление доступа имеют самый низкий приоритет, а запрещение доступа — самый высокий. Это значит, что доступ к данным может быть получен только явным его предоставлением при отсутствии запрещения доступа на любом другом уровне иерархии системы безопасности. Если доступ явно не предоставлен, пользователь не сможет работать с данными.

Реализация фильтрации на уровне строк

Фильтрация на уровне строк используется для приложений, которые хранят сведения в одной таблице. Для реализации фильтрации на уровне строк в каждой строке предусмотрен столбец, в котором задается отличительный параметр. Вы создаете политику безопасности или представление на основе этой таблицы для фильтрации строк, к которым пользователь имеет доступ. Затем вы создаете параметризованные хранимые процедуры, контролирующие типы запросов, которые может выполнять пользователь.

Естественным следствием развития СУБД является проблема организации совместной работы нескольких пользователей с одной и той же совокупностью данных, или, кратко, проблемы многопользовательского доступа к данным.

С точки зрения организации совместного доступа к данным со стороны нескольких пользователей режимы работы с ними делятся на режим монопольного (эксклюзивного) доступа и режим общего (разделенного) доступа.

Режим монопольного доступа к базе данных предусматривает, что только один из пользователей (программных процессов) может работать с ней, а возможность ее открытия другими пользователями (процессами) блокируется. Открытие базы данных в монопольном режиме, используется для выполнения операций по изменению структуры таблиц и связей между ними, экспорта большого количества информации, выполнения служебных операций с данными (сохранение, восстановление, сжатие) и т. п.

Соответственно, в режиме разделенного доступа сразу несколько пользователей могут работать с базой данных. Для предотвращения возможных конфликтов при попытках со стороны различных пользователей изменить одни и те же записи в СУБД используется механизм блокировок. Механизм блокировок предоставляет более гибкие возможности для манипуляций с данными по сравнению с режимом монопольного доступа.

- Отсутствует - допускается одновременное изменение записей со стороны нескольких пользователей. При этом если два пользователя пытаются сохранить произведенные изменения в одной и той же записи, то второму пользователю выводится предупреждающее сообщение, на основе которого

он может либо отказаться от дальнейших действий, либо заместить изменения, сделанные первым пользователем, сохранив собственный вариант.

- Всех записей - происходит блокировка всех записей в источнике данных при его открытии одним из пользователей, в результате чего он может беспрепятственно изменять его. Другие пользователи имеют доступ только на чтение (просмотр).

- Изменяемой записи - один из пользователей получает доступ на изменение нужной ему записи, а другие пользователи могут только читать содержащиеся в ней данные. Данный режим накладывает минимальные ограничения на совместную работу.

Другим существенным вопросом, который должен быть решен для обеспечения нормального функционирования многопользовательских СУБД, является организация системы администрирования данных. Среди задач администрирования могут быть названы:

- создание системы пользователей и разделение прав доступа различных пользователей к объектам СУБД;

- организация и поддержание системы резервного хранения информации и ее восстановления в случае программных и аппаратных сбоев;

- мониторинг программных и аппаратных ресурсов, задействованных для обеспечения работы СУБД, и принятие на его основе решений по оптимизации их использования.

Привилегии могут передаваться субъектам (отдельным пользователям), группам, ролям или всем пользователям.

Группа - это именованная совокупность пользователей. Объединение субъектов в группы облегчает администрирование баз данных и, как правило, строится на основе формальной или фактической структуры организации. Каждый пользователь может входить в несколько групп. Когда пользователь тем или иным способом инициирует сеанс работы с базой данных, он может указать, от имени какой из своих групп он выступает.

Роль - это еще один возможный именованный носитель привилегий. С ролью не ассоциируют перечень допустимых пользователей - вместо этого роли защищают паролями.

Привилегии роли имеют приоритет над привилегиями пользователей и групп. Иными словами, пользователю как субъекту не обязательно иметь права доступа к объектам, обрабатываемым приложениям с определенной ролью.

24. Галерея компонентов

Галерея компонентов является хранилищем любых объектов VisualFoxPro(проекты, библиотеки, классы, формы, отчеты, кнопки и др.). Компоненты Галереи можно переносить на форму, и наоборот.

Галерея компонентов является средством организации разработки в VisualFoxPro, позволяющим разработчику группировать и упорядочивать компоненты, такие как используемые в работе проекты, библиотеки, классы, формы, отчеты, кнопки и т. п. Созданное вами упорядочение можно динамически изменять, что позволит использовать разные подходы к классификации компонентов разработки.

В Галерее компонентов для работы можно разместить различные элементы Visual FoxPro, локальные и удаленные документы, файлы или папки, Automation-серверы, например, Microsoft Excel, Microsoft Word и HTML-файлы. В ней содержатся также новые базовые классы Visual FoxPro.

Галерея компонентов призвана создать рабочую среду и являться хранилищем данных, используемым для быстрой разработки приложений. В ней можно разместить не только компоненты, из которых создается приложение, но и созданные вами или сторонними организациями проекты, статьи и другие документы, содержащие информацию, полезную для разработки. Вокне Галереи компонентов можно также хранить ссылки на используемые при разработке Web-страницы.

Галерея компонентов располагает средствами для создания новых проектов, форм, а также для изменения свойств объектов и классов. Объекты, размещенные в Галерее компонентов, можно переносить в проекты и формы (и наоборот) с помощью метода "перенести-и-оставить". (drag n drop)

ComponentGallery(Галерея компонентов) – мощное средство VisualFoxPro7.0 для быстрой разработки приложений с использованием самых различных содержащихся в ней локальных и удаленных компонентов – библиотек и классов, готовых проектов, текстовых документов, ссылок на Web-страницы, форм, отчетов и т.д. Для работы с ComponentGallery следует выполнить следующую команду меню Tools-ComponentGallery. В открывшемся окне в левой части представлен список стандартных каталогов VisualFoxPro, а в правой – содержимое выбранного каталога. Окно ComponentGallery используется по умолчанию и соответствует режиму Default. Кроме этого в ниспадающем списке можно выбрать режим ClassbyType – для вывода перечня классов, сгруппировав их по типу, режим BuildingApps – для вывода перечня класса используемых при построении приложения, режим FastForm – для вывода перечня классов с заготовками для быстрого построения формы.

В левой части окна содержится дерево каталогов, а в правой – содержимое каталога.

Для настройки Галереи есть кнопка **Options** (рисунок 2.16.4.2).

Страница Standard имеет следующие параметры:

Enabled item renaming - разрешение менять имена каталогов и объектов;

FFC Builder Lock - запуск строителя объекта при его размещении;

Drag and drop... - перенос объектов из Галереи на Рабочий стол FoxPro;

Advanced... - добавление страниц Type, Scripts, Views и Comments;

Modify/Run... - корректировка/выполнение объекта при двойном щелчке.


Страница Standard содержит список каталогов и кнопку **New** для включения нового каталога. Имеется список стандартных каталогов: **Visual FoxPro Catalog** (базовые классы, шаблоны и др.), **Favorites** (часто используемые объекты), **My Base Catalog** (подклассы базовых классов), **ActiveX Catalog** (зарегистрированные ActiveX-объекты), **World Wide Web** (список адресов Web-страниц), **Multimedia Catalog** (рисунки, звуки), **Visual FoxPro Samples** (примеры приложений).

Страница Dynamic View содержит список представлений и кнопки **New**, **Edit** и **Remove** - добавление, изменение и удаление представления.

Средство **Class Browser**

В Visual FoxPro имеется удобное средство, предназначенное для работы с классами, — это **Class Browser** (Обзор классов), с помощью которого вы можете просматривать библиотеки классов, создавать новые классы и редактировать существующие.

Для открытия окна **Class Browser** (Обзор классов) выполните одно из следующих действий:


- ☐ в меню **Tools** (Сервис) выберите команду **Class Browser** (Обзор классов);
- ☐ нажмите на стандартной панели инструментов кнопку  **Class Browser** (Обзор классов);
- ☐ в командном окне выполните команду `DO (_BROWSER)`.


Независимо от выбранного вами способа, будет запущено приложение **Browser.app**, которое открывает окно **Class Browser** (Обзор классов) (рис. 19.28), содержащее панель инструментов для управления классами и список объектов выбранной библиотеки.

После открытия библиотеки классов в левой части окна **Class Browser** (Обзор классов) будет отображен иерархический список классов выбранной библиотеки со значками, которые вы присвоили с помощью команды **Class Info** (Информация о классе) из меню **Class** (Класс). В правой части окна отображается список объектов, входящих в данный класс, его свойства и методы. В нижней части окна содержатся параметры выбранного класса или элемента класса.

ЗАМЕЧАНИЕ

Двойной щелчок на имени класса в левой области окна **Class Browser** (Обзор классов) автоматически открывает данный класс в конструкторе классов.

В окне **Class Browser** (Обзор классов) могут отображаться несколько библиотек классов. Для добавления новой библиотеки нажмите кнопку **View Additional File** (Просмотр дополнительного файла)  на панели инструментов окна **Class Browser** (Обзор классов) и в диалоговом окне **Open** (Открыть) выберите библиотеку классов.

Для просмотра любой другой библиотеки классов нажмите кнопку **Open** (Открыть)  на панели инструментов и в диалоговом окне **Open** (Открыть) выберите требуемую библиотеку.

ЗАМЕЧАНИЕ

В окне **Class Browser** (Обзор классов) могут отображаться не только библиотеки классов, но и формы.

Чтобы просмотреть классы одного типа (например, класс кнопок), выберите из раскрывающегося списка **Class Type** (Тип класса) наименование требуемого типа.

Управление классами

Для создания нового класса, изменения его имени, удаления его из библиотеки классов вы можете в окне **Class Browser** (Обзор классов) использовать кнопки, описанные в табл. 19.10.

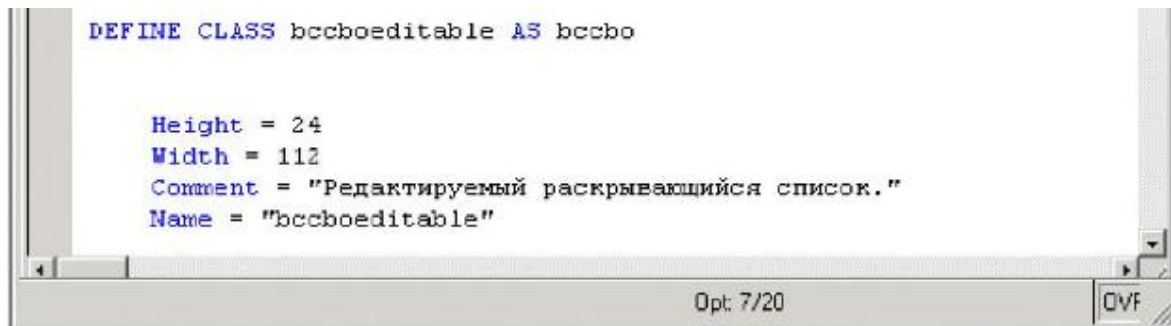


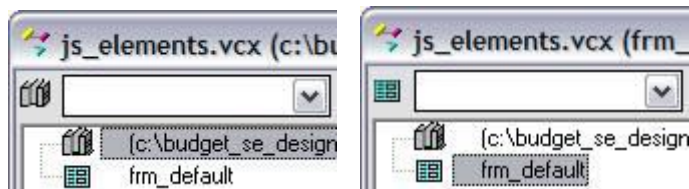
Рис. 19.29. Исходный текст описания класса

Class Browser отображает классы в библиотеках классов или формах, а также информацию библиотеки типов в файлах библиотек типов (.tlb), библиотек объектов (.olb) или исполнимых файлах (.exe). Вы можете использовать Class Browser для просмотра, использования и управления классами и их, определенных пользователями, членов. Для получения более подробной информации смотрите описание [Работа с Class Browser](#).

В Visual FoxPro 9.0, вы можете, кроме того, отобразить определения классов в программных файлах (.prg.). Многие возможности Class Browser, которые доступны для библиотек классов доступны и для программных файлов. Вы можете выполнить двойной щелчок на методе или событии, для редактирования содержащегося в нем кода, и вы можете выполнить двойной щелчок на свойстве отображения его значения. Вы можете, кроме того, выполнить двойной щелчок в программном классе для изменения его структуры. Классы в программных файлах могут быть просмотрены в иерархическом или неиерархическом виде, и вы можете отфильтровывать защищенные и скрытые члены.

Иконка класса

Отображает иконку, представляющую выбранный класс или библиотеку классов. Эта иконка появляется рядом с боксом типа только того, когда класс выбран в списке классов Class Browser. Вы можете указать другую иконку (.ico) или файл bitmap (.bmp) для использования с классом.



Для получения более подробной информации смотрите описание [Как: указать внешний вид для классов в Design-Time](#).

Бокс выбора типов

Позволяет производить выбор или ввод типа класса или символьной строки для фильтрации перемещений. Выпадающий список показывает базовые классы. Список, кроме того, сохраняет историю типов класса и фильтры которые вы выбрали или ввели для текущего экземпляра Class Browser.

Командные кнопки Class Browser

Выполняют различные команды Class Browser. Контекстное меню предоставляет дополнительную функциональность.



Для получения более подробной информации смотрите описание [Командные кнопки Class Browser](#).

Список классов

Отображает классы и субклассы в открытом файле библиотек классов (.vcx), формы (.scx) или программы (.prg).

Замечание

Символ шеврона (<<), появляющийся рядом с классом, указывает на то, что родительский класс размещен в файле, который не отображается в списке классов.

Список членов

Отображает объекты-члены и любые, определенные пользователем, свойства и методы класса или формы, выбранного в списке классов. Вы можете отфильтровать этот список выбором одной из приведенных ниже опций из контекстного меню списка членов:

- **Protected filter** Если иконка защиты (в виде ключа) отображается рядом с элементом контекстного меню Protected filter, защищенные члены
 - отображаются в списке членов. Защищенные члены, кроме того, указываются с помощью звездочки (*).
- **Hidden filter** Если иконка сокрытия (в виде замка) отображается рядом с элементом контекстного меню Hidden filter, скрытые члены отображаются в списке членов. Скрытые члены, кроме того, указываются с помощью крышки (^).
- **Empty filter** Если значок отметки отображается рядом с элементом контекстного меню Empty filter, то пустые методы отображаются в списке членов. Пустые методы указываются также с помощью символа тильды (~).

Бокс описания класса

Отображает описание выбранного класса. Это бокс размещается слева внизу окна Class Browser. Вы можете отредактировать описание в этом боксе.

Бокс описания члена

Отображает информацию о выбранном члене класса. Это бокс размещается справа внизу окна Class Browser.

Для объектов-членов, содержимое бокса маркировано, как "только для чтения" и отображает информацию о классе и базовом классе. Для членов, представляющих собой пользовательские свойства или методы, бокс отображает описание, которое вы можете редактировать. Для экземпляров класса, бокс отображает ("только для чтения") описание, которое включает в себя область видимости переменной (public или hidden), имена членов и значения свойств.

Использование браузера объектов

В Visual FoxPro 7 появилось новое средство для просмотра списка классов, свойств, методов, событий и констант библиотек COM-объектов или ActiveX-компонентов — Object Browser (Браузер объектов). В окне браузера можно выполнять поиск заданного значения в пределах всей библиотеки.

Для запуска браузера можно воспользоваться одним из следующих способов.

- Выполнить команду Object Browser (Браузер объектов) из меню Tools (Сервис).
- Нажать кнопку Object Browser на стандартной панели инструментов.

Для открытия библиотеки и просмотра ее структуры в окне браузера объектов предназначена кнопка Open Type Library (Открыть библиотеку) При нажатии этой кнопки открывается диалоговое окно Open (Открыть), которое содержит три вкладки следующего назначения:

- Recent History (Хронология) — содержит список открытых ранее библиотек;
- COM Libraries (COM-библиотеки) — содержит список COM-библиотек, имеющихся на компьютере;
- Current Selection (Выбранные) — содержит список открытых в окне Object Browser (Браузер объектов) библиотек.

При нажатии кнопки Find (Найти) в верхней части окна Object Browser (Браузер объектов) появляется поле Look for (Значение для поиска) и кнопка Go (Начать поиск), предназначенные для текстового поиска.

26. Организация помощи и подсказки в VFP.

Для отображения справочной информации в Visual FoxPro вы можете использовать строку состояния, диалоговое окно и справочную систему в формате WinHelp или HTML.

Строка состояния

В Visual FoxPro строка состояния используется для пояснения назначения элементов управления формы и пунктов меню. Пояснения к элементам управления, отображаемые в строке состояния, задаются в свойстве statusBarText (Текст строки состояния). 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 ToolTipText (Текст подсказки).

Использование диалоговых окон

В процессе выполнения приложения осуществляются разнообразные проверки, например, проверка соответствия введенного в поле значения заданным условиям. Если в результате проверки окажется, что введенное значение не удовлетворяет требуемым условиям, необходимо об этом сообщить пользователю. Из текста сообщения должна быть ясна причина ошибки. Кроме того, текст должен содержать рекомендации по ее исправлению. Для вывода информационного сообщения вы можете воспользоваться функцией MESSAGEBOX или создать собственную функцию. Аргумент текст сообщения содержит текст выводимого сообщения. Длинное сообщение автоматически располагается на нескольких строках. Если сообщение состоит из нескольких предложений и вы хотите разбить его на абзацы, то используйте функцию снк(13) для формирования новой строки.

Аргумент тип Диалогового Окна задает перечень кнопок и значков, которые будут отображаться в диалоговом окне. Данный аргумент является числовым и его значение определяется как сумма трех параметров: типа кнопок, типа значков и номера кнопки, выбранной по умолчанию. Возможные значения этих параметров приведены в табл. 12.1—12.3.

Создание справочной системы в формате WinHelp

Основой справочной системы являются темы, содержащие пояснительный текст. Содержание справочной системы включает список тем, доступных в системе. Каждая тема имеет заголовок и уникальный символьный идентификатор. Дополнительно каждой теме можно поставить в соответствие уникальный индекс темы, который должен быть целым числом.

В справочной системе для поиска темы используются ключи, содержащие название темы и ссылку на нее. Каждая тема может иметь более одного ключа поиска. Кроме того, один ключ может содержать ссылку на несколько тем.

Для организации контекстного вызова темы из справочной системы вы можете использовать числовые значения индексов или значения ключа. Использование идентификаторов тем для контекстного вывода справочной информации не допускается.

Чтобы связать между собой отдельные темы, используются перекрестные ссылки. При этом текст, используемый для организации перекрестной ссылки, выделяется зеленым цветом и подчеркиванием.

В описании любой системы используются термины, специфичные для конкретной системы. Например, в системах складского учета такими терминами будут накладная, счет, отпуск товара. В качестве термина может рассматриваться не только отдельное слово, но и любая фраза из текста темы. Справочная система Windows позволяет дать каждому термину приложения краткое определение. Такие термины на экране выделены зеленым цветом и пунктирным подчеркиванием. Если щелкнуть мышью на термине, для которого определено краткое описание, это описание появится на экране в рамке поверх текста темы.

Создание справочной системы в формате WinHelp включает следующие этапы:

1. Создание описания справочной системы. Для этого можно использовать любой текстовый редактор, поддерживающий формат RTF. Описание содержит темы, индексы и ключи справочной системы. Структура документа должна соответствовать требованиям, предъявляемым компилятором справочной системы (Help Compiler).
2. Формирование файла с расширением CNT, содержащего иерархическое содержание создаваемой справочной системы.
3. Формирование текстового файла с расширением HPI, содержащего параметры для компиляции справочной системы: имя файла с описанием справочной системы, заголовки справочной системы.
4. Компиляция созданного текстового описания с использованием Help Compiler.
5. Определение свойства HelpContextid для требуемых объектов формы, а также создание

27. События в VFP.

Событие	Возникает
Activate	При активизации объектов FormSet (Набор форм), Form (Форма), Page (Вкладка формы) или при отображении объекта Toolbar (Панель управления)
AfterBuild	При перестроении проекта или приложения, а также при создании динамической библиотеки (dll) или выполняемого файла (exe) на основе проекта
AfterCloseTables	После закрытия таблиц или представлений данных
AfterDock	После того, как фиксируется панель инструментов
AfterRowColChange	При переходе в другую строку или другой столбец объекта Grid (Таблица)
BeforeDock	Имеет место перед фиксацией объекта Toolbar (Панель управления)
BeforeOpenTables	Перед открытием таблиц и представлений, связанных со средой данных формы, набора форм или отчета
BeforeRowColChange	Перед тем, как пользователь изменяет активную строку или столбец объекта Grid (Таблица)
6. Click	При нажатии и отпускании левой кнопки мыши

7.	CommandTargetExec	При активизации программой-контейнером приложения типа Active Document (Активный документ)
	ConrmandTargetQuery	При изменении программой-контейнером пользовательского интерфейса
	ContainerRelease	При закрытии программой-контейнером приложения типа Active Document (Активный документ)
	DbClick	При двойном нажатии и отпуске левой кнопки мыши
	Deactivate	Когда деактивируется объект типа форма
	Deleted	Когда пользователь удаляет пометку записи на удаление или когда выдается команда DELETE
	Destroy	Когда освобождается объект
	DownClick	При нажатии кнопки со стрелкой, направленной вниз на объектах типа comboBox (Раскрывающийся список), ListBox (Список) и Spinner (Счетчик)
	DragDrop	После завершения операция "перенести-и-оставить"
	DragOver	Когда элемент управления, переносимый с помощью мыши, накрывает объект назначения
	Message	Отображает сообщение в строке состояния, когда объект получает управление
	MiddleClick	При нажатии средней кнопки мыши (для трехкнопочной мыши)
	MouseDown	При нажатии кнопки мыши
	MouseMove	При перемещении указателя мыши по объекту
	MouseUp	При отпуске кнопки мыши
	MouseWheel	При вращении колесика мыши, если мышь его содержит
	Moved	При перемещении объекта на новое место или когда значения свойства Top или Left объекта-контейнера изменяются программным способом
	OLECompleteDrag	Когда данные помещены на новое место или отменена операция "перенести-и-оставить"
	OLEDragOver	Возникает в том случае, если свойство OLEDropMode имеет значение 1 и данные перенесены от источника к приемнику
	OLEGiveFeedBack	После каждого события OLEDragOver
8.	OLESetData	Возникает у источника данных при выполнении метода GetData, если нет данных в нужном формате

Событие	Возникает
DropDown	В момент, когда после нажатия кнопки со стрелкой в элементе управления ComboBox (Раскрывающийся список) должен появиться список
Error	При возникновении ошибки
ErrorMessage	Используется для определения сообщения об ошибке, когда событие Valid возвращает значение False
GotFocus	Когда объект получает фокус в результате действий пользователя или выполнения программного кода
HideDoc	При переходе из активного документа
Init	При создании объекта, когда объект еще не выведен на экран
Interactive Change	При изменении значения элемента управления с помощью клавиатуры или мыши
KeyPress	При нажатии и отпуске клавиши
Load	Имеет место непосредственно перед созданием объекта
LostFocus	Когда объект теряет фокус

Событие	Возникает
OLEStartDrag	В начале переноса данных с помощью механизма "перенести-и-оставить"
Paint	При перерисовывании формы или панели инструментов
ProgrammaticChange	При изменении в коде значения элемента управления
QueryAddFile	Перед добавлением файла в проект
QueryModifyFile	Перед изменением файла в проекте
QueryRemoveFile	Перед удалением файла из проекта
QueryRunFile	Перед выполнением файла или предварительным просмотром отчета или почтовой этикетки
QueryUnload	Перед выгрузкой объекта Form (Форма)
RangeHigh	Возвращает наибольшее значение счетчика или число элементов списка и имеет место, когда элемент управления Spinner (Счетчик) или TextBox (Поле ввода) теряет фокус и когда элемент управления ComboBox (Раскрывающийся список) или ListBox (Список) получает фокус
9. RangeLow	Возвращает наименьшее значение счетчика или значение первого элемента списка и имеет место, когда элемент управления Spinner (Счетчик) или TextBox (Поле ввода) теряет фокус или когда элемент управления ComboBox (Раскрывающийся список) или ListBox (Список) получает фокус
ReadActivate	При активизации новой формы в наборе форм
ReadDeactivate	При деактивизации формы в наборе форм
ReadShow	Когда выдана команда SHOW GETS, В активном объекте FormSet (Набор форм) активизируется объект FormSet (Набор форм)
ReadValid	Имеет место после деактивизации объекта FormSet (Группа форм)
ReadWhen	Имеет место после загрузки объекта FormSet (Набор форм)
Resize	При изменении размеров объекта
RightClick	При нажатии и отпуске правой кнопки мыши
Run	Имеет место, когда Active Document (Активный документ) готов к выполнению кода пользователя
Scrolled	При прокрутке данных в объекте управления Grid (Таблица)

Событие	Возникает
ShowDoc	При переходе в активный документ
Timer	Когда истекает интервал времени, заданный свойством Interval (Интервал)
UIEnable	Имеет место для всех объектов, содержащихся внутри объекта Page (Вкладка формы), каждый раз когда объект Page (Вкладка формы) активизируется или деактивизируется
UnDock	При перемещении объекта ToolBar (Панель управления) с помощью мыши
Unload	При освобождении объекта
UpClick	При нажатии кнопки с направленной вверх стрелкой на объектах управления типа ComboBox (Раскрывающийся список), ListBox (Список) и Spinner (Счетчик)
Valid	Перед тем, как элемент управления теряет фокус
When	Перед тем, как элемент управления получает фокус

28. Транзакции Visual FoxPro

Транзакция (англ. transaction) — группа последовательных операций с базой данных, которая представляет собой логическую единицу работы с данными. Транзакция может быть выполнена либо целиком и успешно, соблюдая целостность данных и независимо от параллельно идущих других транзакций, либо не выполнена вообще и тогда она не должна произвести никакого эффекта. Транзакции обрабатываются транзакционными системами, в процессе работы которых создаётся история транзакций.

Различают последовательные (обычные), параллельные и распределённые транзакции. Распределённые транзакции подразумевают использование более чем одной транзакционной системы и требуют намного более сложной логики (например, two-phase commit — двухфазный протокол фиксации транзакции). Также в некоторых системах реализованы автономные транзакции, или под-транзакции, которые являются автономной частью родительской транзакции.

Пример транзакции:

Необходимо перевести с банковского счёта номер 5 на счёт номер 7 сумму в 10 денежных единиц. Этого можно достичь, к примеру, приведённой последовательностью действий:

- Начать транзакцию
прочитать баланс на счёту номер 5
уменьшить баланс на 10 денежных единиц
сохранить новый баланс счёта номер 5
прочитать баланс на счёту номер 7
увеличить баланс на 10 денежных единиц
сохранить новый баланс счёта номер 7
- Окончить транзакцию

Транзакции в FoxPro.

BEGIN TRANSACTION – открывает блок транзакций

END TRANSACTION – закрывает блок транзакций, сохраняет результат

Транзакции имеют эксклюзивный доступ, поэтому пользователи сети не имеют доступ к записям, использующимся в блоке транзакций.

В блоке транзакций невозможно: добавлять таблицу; модифицировать, создавать, удалять базы/процедуры/соединения/локальные представления; чистить и закрывать таблицы

29. Конструктор базы данных

Конструктор баз данных отображает все таблицы, представления и отношения, содержащиеся в базе данных. Когда окно конструктора баз данных активно, Visual FoxPro отображает меню Database и панель инструментов Database Designer.

Каждая таблица представлена окном с варьируемыми размерами, в котором перечисляются поля таблицы и ее индексы, если таковые существуют. Конструктор баз данных показывает установленные меж-таблицные отношения в виде линий, соединяющих индексы этих таблиц.

Вы можете установить постоянные отношения между таблицами базы данных.

► Как создать постоянные отношения между таблицами

- В конструкторе баз данных выберите индекс, который вы хотите связать, и переместите его на индекс связываемой таблицы.

- Или -

- Включите предложение FOREIGN KEY в команду CREATE TABLE или команду ALTER TABLE.

► Как удалить постоянное отношение между таблицами

1. В конструкторе баз данных щелкните мышью линию, соединяющую таблицы.

Толщина линии увеличится, отмечая выбор отношения между таблицами.

2. Нажмите клавишу delete.

- Или -

- Включите предложение DROP FOREIGN KEY в команду ALTER TABLE.

Отношения, создаваемые в базе данных, называются постоянными, потому что они хранятся как часть базы данных. Каждый раз, когда таблица используется в конструкторе окружения данных при создании форм или в конструкторе запросов или представлений, эти постоянные отношения возникают как определенные связи или объединения таблиц по умолчанию.

Если таблицы еще не имеют индексов, необходимо открыть эти таблицы в конструкторе таблиц и добавить к ним индексы.

► Как создать отношение между таблицами

- Перетащите указатель из одного индекса таблицы в соответствующий индекс другой таблицы.

Замечание Индекс родительской таблицы (таблицы из которой он перетаскивается) должен быть первичным или индексом-кандидатом.

Как только вы установили отношение, оно показывается в конструкторе баз данных в виде линии, соединяющей две таблицы.

Возможно, вам захочется установить отношения для таблиц базы данных. Для создания отношения необходимо, чтобы у связываемых таблиц имелись индексы.

Чтобы выполнить подготовительные процедуры для создания отношения, найдите общее поле в двух таблицах и добавьте индексы. Например, обе таблицы Customer и Orders имеют поле Cust_ID, потому что один покупатель может сделать несколько заказов. Чтобы установить отношение один-ко-многим между ними, необходимо добавить к таблице Customer первичный индекс поля Cust_ID, а к таблице Orders — обычный индекс поля Cust_ID данной таблицы.

30. Основные этапы построения БД.

Процесс создания базы данных следует тщательно продумать, поскольку допущенные ошибки исправлять намного сложнее, когда база данных наполнена информацией. Разработку базы данных лучше выполнять в несколько этапов.

Постановка задачи. В процессе разработки баз данных обычно присутствуют две стороны: заказчик и разработчик. Заказчик определяет цель и задачи создания базы данных, а разработчик реализует эту задачу. В диалоге между заказчиком и разработчиком нужно решить следующие вопросы:

- для чего создается база данных;
- какие данные будут храниться;
- как нужно обрабатывать данные и какие результаты при этом нужно получить.

Если база данных небольшая, то заказчик и разработчик может быть одним и тем же лицом. Однако и в этом случае перед началом создания базы данных следует дать подробные ответы на приведенные выше вопросы.

Разработка структуры базы данных. На этом этапе следует определить, какие таблицы будут использоваться в базе данных и как они будут связаны между собой. Результат разработки структуры можно представить в виде схемы данных (см. рис. 7.14), которую можно начертить на листе бумаги. Это самый важный этап при создании базы данных, поскольку ошибки в структуре могут стать источником серьезных проблем при последующем выполнении обработки данных, а изменение структуры уже готовой базы данных — относительно сложная задача. Считается, что правильная структура базы данных — это половина ее успеха, поэтому созданную структуру нужно тщательно проанализировать и при необходимости внести в нее изменения.

Для разработки структуры небольшой базы данных из нескольких таблиц самыми подходящими инструментами могут оказаться лист бумаги и карандаш. Для проектирования объемных баз данных

существуют специальные программные средства, для работы с которыми необходим соответствующий уровень подготовки.

Создание таблиц и связей между ними. После разработки структуры базы данных можно приступить к созданию самой базы данных, таблиц и связей между ними. Для создания таблицы нужно иметь следующую информацию:

- как будут называться все поля таблицы и какой тип данных будет использоваться для каждого поля;
- какие поля в таблице будут являться ключевыми (уникальными);
- какие дополнительные свойства полей будут применяться.

Создание форм, запросов и отчетов. Эти объекты создаются для обработки данных, вывода результатов и удобства работы пользователя. Их создание будет подробно рассмотрено в следующих уроках.

ПРИМЕЧАНИЕ

В Access имеются шаблоны нескольких баз данных, с помощью которых можно быстро создать новую базу данных с готовыми таблицами, формами и отчетами, не углубляясь в параметры создаваемых объектов. Этот способ доступен пользователям с минимальным уровнем знаний, однако имеет существенный недостаток: такая база данных часто не удовлетворяет всем поставленным требованиям и ее приходится доделывать вручную. Чтобы лучше разобраться с принципами построения баз данных, далее будет рассмотрено создание базы данных «с нуля». Полученные знания позволят редактировать любые базы данных, в том числе и созданные на основе встроенных шаблонов.

Техническое задание на разработку учебной базы данных

Необходимо создать базу данных для диспетчера такси, которая должна отвечать следующим требованиям.

-В эту базу диспетчер должен заносить поступающие вызовы клиентов и регистрировать их исполнение.

-На связи с диспетчером находятся несколько автомобилей, каждый из них закреплен за определенным водителем и имеет бортовой номер. При поступлении вызова диспетчер назначает автомобиль для выполнения заказа и фиксирует информацию о заказе в базе данных.

-В базе данных нужно вести учет заказов постоянных клиентов, каждый из которых имеет карточку с персональным номером. Кроме номера карточки база данных должна позволять заносить фамилии, адреса и телефоны клиентов.

-Кроме учета оперативной информации база данных должна обеспечивать получение различных итоговых данных, например количество и сумма заказов постоянных клиентов, интенсивность использования автомобилей и др.

Поставив задачу, можно приступить к разработке структуры базы данных.

31. Навигационные команды

В Visual FoxPro доступ к данным осуществляется с помощью указателей. Использование указателей подразумевает, что для обработки конкретных данных в таблицах сначала по указателю находят необходимую запись, затем выполняют над ней нужные команды и, наконец, переходят к следующей записи. Команды, использующие принцип перемещения по указателям, называются «навигационными».

Перемещение по таблице

Команды

GO [RECORD] nRecordNumber [IN nWorkArea | IN cTableAlias]

GO TOP | BOTTOM [IN nWorkArea | IN cTableAlias]

GOTO [RECORD] nRecordNumber [IN nWorkArea | IN cTableAlias]

GOTO TOP | BOTTOM [IN nWorkArea | IN cTableAlias]

Перемещают файловый указатель таблицы (либо текущей, либо указанной в nWorkArea | IN cTableAlias) на запись с заданным номером (nRecordNumber) или в начало (GO TOP) или в конец (GO BOTTOM) таблицы.

RECORD nRecordNumber – указывает на физический номер записи. Если номер превысит количество записей в таблице, то будет сгенерирована ошибка “Record is out of range».

TOP - устанавливает указатель таблицы на первую запись. Если используется возрастающий индекс, то первой записью будет являться запись с наименьшим значением ключа, если убывающий – с наибольшим.

GO BOTTOM – устанавливает указатель на последнюю запись. Если используется возрастающий индекс, то последней является запись с наибольшим значением ключа.

Команда

SKIP [nRecords] [IN WorkArea | cTableAlias]

Перемещает указатель по таблице вперёд или назад.

nRecords – количество записей, на которое перемещается указатель. Если число опущено, то указатель перемещается на следующую запись. Если число > 0, то перемещение следует к концу файла, если < 0, то к началу.

Добавление записей в таблицу

Добавить записи в таблицу можно различными способами. Сразу после создания таблицы FoxPro предлагает ввести данные вручную. Если отложить ввод данных на более позднее время, вы всегда сможете открыть таблицу в командном окне и дать команду Append

APPEND [BLANK] [IN nWorkArea | cTableAlias] [NOMENU]

Опции:

BLANK – добавляет одну пустую запись в конец таблицы;

IN nWorkArea | cTableAlias – для команды Append без опции BLANK работа происходит с текущей таблицей. Для APPEND BLANK запись добавляется в таблицу, которая указывается в опции;

NOMENU – удаляет меню Table из главного меню VFP. Не позволяет работать с записями таблицы из главного меню.

Если открыть таблицу и просматривая её, дать команду BROWSE, в главном меню FoxPro появляется пункт Table, в нём можно выбрать подпункт Append New Record, при этом в таблицу будет добавлена пустая запись. Если ваша таблица содержит поле типа Integer (AutoInc), то при добавлении записи она сразу же нумеруется.

Хотя для ввода пустой записи есть способ попроще: установить указатель записи на последнюю запись и нажать комбинацию <Ctrl>+<Y>. Пустая запись будет добавлена, станет возможно её заполнить.

Физически запись всегда добавляется в конец файла. Нужный порядок следования обеспечивается главным индексом.

Теперь рассмотрим программные способы добавления записей в таблицу.

Самый простой способ состоит в том, что в таблицу добавляется пустая запись с командой APPEND BLANK, а потом на экране выводится форма для её заполнения.

USE SOTR

APPEND BLANK

DO FORM app_sotr

При этом редактируются поля уже введённой в таблицу записи.

Команда

APPEND FROM ARRAY ArrayName [FOR iExpression]

Добавляет записи в таблицу из массива ArrayName. Количество добавляемых записей равно количеству строк массива. Если количество элементов строки массива больше чем количество полей в таблице, лишние элементы отбрасываются без дополнительных сообщений. Если же количество элементов массива меньше, чем количество полей в таблице, лишние поля заполняются значениями по умолчанию.

Поля типа Memo, Blob и General командой APPEND FROM ARRAY игнорируются.

Записи в эти поля можно производить командой GATHER или APPEND GENERAL.

Добавить запись в таблицу, имеющую поле Integer [AutoInc], возможно если установить предварительно SET AUTOINCERROR = OFF или установить

CURSORSETPRQP ("AutoIncError", . F., "Sotr")

С появлением представлений (view) программисты всё чаще используют их для добавления записей в таблицу. В этом случае создаётся представление (view), созданное View, добавляется в DataEnvironment создаваемой формы. Туда же добавляется и таблицы – источники, из которых создано View. Там же, в DataEnvironment формы, для добавленного Local View устанавливается свойство NoDataOnLoad = .т. Эта настройка означает, что Local View откроется, но без содержимого, только структура. В событии Init формы пишется такой код:

```
Local MyVar
```

```
MyVar = 1
```

```
=ReQuery ("MyView")
```

Этот код наполняет содержимым Local View.

В самой форме располагаются нужные объекты и в их свойстве ControlSource указывается ссылка на поля Local View.

Для сохранения внесённых изменений нужно дать команду TableUpdate ().

32. NOSQL базы данных для Web – приложений

NoSQL в информатике — термин, обозначающий ряд подходов, направленных на реализацию хранилищ баз данных, имеющих существенные отличия от моделей, используемых в традиционных реляционных СУБД с доступом к данным средствами языка SQL. Применяется к базам данных, в которых делается попытка решить проблемы масштабируемости (англ. scalability) и доступности (англ. availability) за счёт атомарности (англ. atomicity) и согласованности данных (англ. consistency).

Когда веб-сайты начали вмещать большие объёмы данных, то резко стала ощущаться низкая производительность в некоторых приложениях при использовании традиционных баз данных, и появилась необходимость в новых решениях. С тех пор развитие баз данных с открытым исходным кодом NoSQL начало набирать свои обороты. Конференция, проходившая в Атланте в 2009 году, считается отправной точкой для начала продвижения баз данных NoSQL.

NoSQL это новое поколение систем управления базами данных, которые отличаются от традиционных систем управления базами данных по нескольким ключевым направлениям. Для начала, базы данных NoSQL не являются реляционными. Это означает, что они не требуют использования соединений и таким образом могут обрабатывать большие объёмы данных, даже в масштабе интернета. Вот почему базы данных NoSQL являются необходимостью для веб приложений, работающих с большими объёмами трафика и генерируемого контента для пользователей. NoSQL является отличным решением для таких сайтов, как Google, Digg, Facebook, Twitter и многих других. Основным преимуществом баз данных NoSQL является их массовая способность масштабирования. Они могут справиться с крайне тяжелой загрузкой и обеспечить максимальную производительность, где даже самая лучшая СУБД, работающая на мощном аппаратном уровне, не сможет за ней угнаться. Другие

преимущества систем управления базами данных NoSQL включает прозрачный шардинг, параллельную обработку, и автоматическое разрешение конфликтов. Базы данных NoSQL очень быстрые, децентрализованы, и очень доступны.

Основные черты

Традиционные СУБД ориентируются на требования ACID к транзакционной системе: атомарность (англ. atomicity), согласованность (англ. consistency), изолированность (англ. isolation), надёжность (англ. durability), тогда как в NoSQL вместо ACID может рассматриваться набор свойств BASE:

- базовая доступность (англ. basic availability) — каждый запрос гарантированно завершается (успешно или безуспешно).
- гибкое состояние (англ. soft state) — состояние системы может изменяться со временем, даже без ввода новых данных, для достижения согласования данных.
- согласованность в конечном счёте (англ. eventual consistency) — данные могут быть некоторое время рассогласованы, но приходят к согласованию через некоторое время.

Термин «BASE» был предложен Эриком Брюером, автором теоремы CAP, согласно которой в распределённых вычислениях можно обеспечить только два из трёх свойств: согласованность данных, доступность или устойчивость к разделению. Разумеется, системы на основе BASE не могут использоваться в любых приложениях: для функционирования биржевых и банковских систем использование транзакций является необходимостью. В то же время, свойства ACID, какими бы желанными они ни были, практически невозможно обеспечить в системах с многомиллионной веб-аудиторией, вроде amazon.com. Таким образом, проектировщики NoSQL-систем жертвуют согласованностью данных ради достижения двух других свойств из теоремы CAP^[4]. Некоторые СУБД, например, Riak, позволяют настраивать требуемые характеристики доступности-согласованности даже для отдельных запросов путём задания количества узлов, необходимых для подтверждения успеха транзакции. Решения NoSQL отличаются не только проектированием с учётом масштабирования. Другими характерными чертами NoSQL-решений являются:

- Применение различных типов хранилищ.
- Возможность разработки базы данных без задания схемы.
- Использование многопроцессорности.
- Линейная масштабируемость (добавление процессоров увеличивает производительность)^[6].
- Инновационность: «не только SQL» открывает много возможностей для хранения и обработки данных.
- Сокращение времени разработки
- Скорость: даже при небольшом количестве данных конечные пользователи могут оценить снижение времени отклика системы с сотен миллисекунд до миллисекунд

Типы хранилищ данных

Описание схемы данных в случае использования NoSQL-решений может осуществляться через использование различных структур данных: хеш-таблиц, деревьев и других.

В зависимости от модели данных и подходов к распределённости и репликации можно выделить четыре типа хранилищ: «ключ-значение» (key-value store), документно-ориентированные (document store), хранилища семейств колонок (column database), графовые базы данных (graph database).

Хранилище «ключ-значение»

Хранилища «ключ-значение» является простейшим хранилищем данных, использующим ключ для доступа к значению. Такие хранилища используются для хранения изображений, создания специализированных файловых систем, в качестве кэшей для объектов, а также в системах,

спроектированных с прицелом на масштабируемость. Примеры таких хранилищ — Berkeley DB, MemcacheDB (англ.)русск., Redis, Riak, Amazon DynamoDB.

Хранилище семейств колонок (или Bigtable-подобные базы данных)

В этом хранилище данные хранятся в виде разреженной матрицы, строки и столбцы которой используются как ключи. Типичным применением этого вида СУБД является веб-индексирование, а также задачи, связанные с большими данными, с пониженными требованиями к согласованности данных. Примерами СУБД данного типа являются: Apache HBase, Apache Cassandra, Apache Accumulo (англ.)русск., Hypertable (англ.)русск., SimpleDB (Amazon.com). Хранилища семейств колонок и документно-ориентированные хранилища имеют близкие сценарии использования: системы управления содержимым, блоги, регистрация событий. Использование отметок времени (timestamp) позволяет использовать этот вид хранилища для организации счётчиков, а также регистрации и обработки различных данных, связанных со временем. Хранилища семейств колонок (англ. column family stores) не следует путать с колоночными хранилищами (англ. column stores). Последние являются реляционными СУБД с раздельным хранением колонок (в отличие от более традиционного построчного хранения данных).

Документно-ориентированная СУБД

Документно-ориентированные СУБД служат для хранения иерархических структур данных. Находят своё применение в системах управления содержимым, издательском деле, документальном поиске и т. п. Примеры СУБД данного типа — CouchDB, Couchbase, MarkLogic, MongoDB, eXist, Berkeley DB XML.

Базы данных на основе графов

Графовые базы данных применяются для задач, в которых данные имеют большое количество связей, например, социальные сети, выявление мошенничества. Примеры: Neo4j, OrientDB, AllegroGraph (англ.)русск., Blazegraph(RDF-хранилище, ранее называлось Bigdata), InfiniteGraph (англ.)русск., FlockDB, Titan. Так как рёбра графа материализованы (англ. materialized), то есть, являются хранимыми, обход графа не требует дополнительных вычислений (как JOIN в SQL), но для нахождения начальной вершины обхода требуется наличие индексов. Графовые базы данных как правило поддерживают ACID, а также имеют различные языки запросов, вроде Gremlin и Cypher (Neo4j).

33. БД MySQL. Основные сведения.

Разработка многостраничных Интернет-сайтов обычно связана с необходимостью хранения и использования больших объемов информации. Удобнее всего представлять данные в форме взаимосвязанных таблиц, работающих под руководством специальных систем управления (СУБД). Разработанная в 1995 году компанией Oracle база данных MySQL предназначена для создания небольших и средних Интернет-проектов. База данных MySQL — это реляционная СУБД, которая может применяться в качестве SQL-сервера.

Преимущества MySQL

Широкая популярность MySQL среди разработчиков обусловлена множеством ее преимуществ:

- отличной скорости работы и обработки данных;
- высокой надежности и стабильности;
- быстрой установке на компьютер;
- функциональности и гибкости;
- безопасности;
- бесплатному распространению;

- открытому коду, позволяющему легко вносить изменения;
- нетребовательности к вычислительным мощностям. Для установки БД подойдет даже компьютер средней производительности, имеющий диск небольшого размера;
- возможности работы с множеством платформ: Linux, Mac OS X, Windows 95/98/NT/2000/XP/Server 2003/Vista/7, FreeBSD, etBSD, OpenBSD, SGI IRIX, Solaris, SunOS, SCO OpenServer, OS/2 Warp, UnixWare, Tru64, WinCE, AIX, BSDi, HP-UX, OpenVMS;
- возможности взаимодействия с интерфейсом программного приложения API и поддержка множества языков программирования, в том числе, популярного PHP, а также C, C++, Delphi, Java, Perl, Python, Ruby, Smalltalk, Паскаль, Tcl и так далее;
- отличной техподдержке;
- постоянному совершенствованию, выпуску новых, улучшенных версий.

Отличные качества MySQL высоко оценены разработчиками во всем мире. В 2011 году она была признана «Лучшей системой управления данными» в конкурсе Impact Awards и в опросе читателей журнала Linux Journal. Более 2000 независимых разработчиков ПО используют ее для создания своих продуктов.

Недостатки MySQL

Наряду с большим количеством достоинств, MySQL не лишена и недостатков. В частности, для работы с мощными корпоративными порталами, на которых представлена самая разнообразная информация, ей не хватает ряда функций:

- возможности применения транзакций;
- возможности использования триггеров для автоматизации контроля за работой БД;
- создания хранимых процедур;
- создания вложенных запросов;
- каскадного обновления данных;
- графического интерфейса пользователя.

Характеристики MySQL

В MySQL версии 3.22 размер таблиц ограничен 4 ГБ, в более поздних версиях он ограничивается максимальным размером файла установленной операционной системы и типом таблицы и в некоторых случаях он может достигать 32 эксабайт. MySQL также имеет ограничение на количество столбцов таблицы — их может быть не более 409.

Версии MySQL

Разработчиками широко используется база данных MySQL версий от 3.2.3 до самых свежих 5.5 и 5.6.

Основные особенности версии 5.5:

- повышение производительности;
- улучшение масштабируемости;
- использование движка InnoDB;
- обновленный механизм репликации;
- совершенствование секционирования данных;
- расширение синтаксиса для фрагментирования объемных таблиц;
- создание новых операций RANGE, LIST;
- новые методы оптимизации;
- совершенствование системы внутренних блокировок;
- встраивание патчей Google с оптимизацией работы InnoDB на многоядерных процессорах.

Новые возможности самой свежей версии 5.6:

- создание интерфейса непосредственного доступа к таблицам InnoDB с использованием API;
- создание в InnoDB индексов для проведения ускоренного поиска по словоформам контента таблиц InnoDB;
- увеличение эффективности оптимизаторов запросов и подзапросов;

- расширение инструментов диагностики работы оптимизатора;
- совершенствование системы диагностики и мониторинга выполнения операций;
- доработка движка InnoDB;
- реализация режима отложенной репликации, позволяющего защитить данные от ошибок оператора;
- расширение максимального размера файлов с логами изменений;
- повышение безопасности доступа и другие.