ГЛАВА 36

Разработка приложений с помощью LiveWire Pro

Майк Морган и Марк Р. Браун

В этой главе...

LiveWire Pro

Как известно, фирма Netscape Communications начала свою деятельность с разработки броузеров и серверов. Компания Netscape имеет огромный опыт по использованию собственных продуктов при разработке Web-страниц. В 1995 году, когда фирма начала расширять номенклатуру выпускаемых серверов, было решено заняться также и созданием средств разработки приложений. Эти средства сейчас известны под названием LiveWire.

Понятие *разработка приложений в Web* подразумевает, что с развитием Web возникли и другие возможности кроме файлов статического HTML, которые ранее применялись на Web-страницах. Все больше разработчиков переходят к использованию интерфейса CGI (Common Gateway Interface), что позволяет им расширять возможности Web-страниц. Однако написание сценария CGI — это довольно сложное дело для непрофессионального программиста, что ограничивает использование преимуществ этой технологии.

Средства разработки, созданные фирмой Netscape, позволяют использовать в приложениях компоненты языка Java и JavaScript тем, у кого нет достаточного опыта программирования. Кроме того, в LiveWire Pro имеются средства, позволяющие интегрировать в Web-страницы базу данных, которая может работать с языком SQL.

Структура и составные части LiveWire Pro

В настоящее время на рынке intranet усиливается соперничество между фирмами Netscape Communications и Microsoft Corporation. Microsoft имеет почти двадцатилетний опыт работы на рынке компьютерных приложений. Биллу Гейтсу удалось собрать команду аналитиков, программистов и менеджеров, которые способны быстро создавать и поддерживать программные продукты. Особенно широко используется корпоративными пользователями операционная система Windows 95 фирмы Microsoft, поэтому обычно именно эта система используется в intranet. Разрабатывая серверы второго поколения и LiveWire на платформе UNIX и Windows NT, фирма Netscape добилась того, что при выборе сервера предприятия имеют возможность использовать и решения, предлагаемые Netscape.

Фирма Netscape Communications была основана в 1994 году. Однако, в отличие от Microsoft, фирма Netscape изначально ориентировалась в своей работе на сети. Главные достижения Netscape связаны с программными продуктами для использования в сети, и особенно в WWW.

За последние годы бума в использовании персональных компьютеров Microsoft и другие фирмы сколотили миллиардные состояния на неграмотности, продавая интерпретаторы компьютерного языка BASIC, которые позволили миллионам непрофессиональных программистов самостоятельно создавать приложения. В борьбе за победу на рынке корпоративных компьютерных сетей Microsoft и Netscape понимают, что победителем будет та компания, которая предлагает лучшее визуальное программное окружение, позволяющее непрофессиональным программистам создавать (зачастую только внешне) сложные приложения для Web.

Microsoft сделала ставку на Visual Basic Script (VBScript) и объекты ActiveX, а Netscape предлагает LiveWire Pro, который входит в комплект поставки сервера Netscape — Enterprise Server Pro 3.0 или SuiteSpot 3.01. В пакет LiveWire Pro входят шесть компонентов.

- Netscape Navigator Gold — программа-клиент Netscape Navigator со встроенными возможностями текстового процессора, которая позволяет пользователям создавать и редактировать документы в среде WYSIWYG.
- LiveWire Site Manager — визуальное средство, позволяющее одновременно просматривать все страницы приложения, а также управлять страницами, ссылками и файлами с помощью "мышиной возни", т.е. метода "перетащить и опустить".
- LiveWire JavaScript Compiler — расширение серверов Netscape, позволяющее создавать коммерческие приложения сервера на языке JavaScript, причем часть обработки выполняется на компьютере клиента, а часть — на сервере.
- Database Connectivity Library — программы, создающие интерфейс API (Application Programmer Interface) для JavaScript и некоторых коммерческих реляционных баз данных.
- Informix-Online Or Oracle7 Workgroup Database — один из трех вариантов: версия разработчика INFORMIX-OnLine Workgroup, версия entry-level сервера On-Line Dynamic Server, или limited-deployment копия Oracle7 Workgroup Server.
- Crystal Reports Professional Version 5.0 — средство для разработки отчетов и анализа данных под управлением Windows, входит в версию LiveWire Pro для Windows NT.

Используя только компоненты LiveWire Pro, можно разработать приложение для доступа и передачи данных в базу данных Informix и представления данных в виде динамических Web-страниц или в качестве потока данных для приложения клиента.

На заметку Более подробную информацию о Crystal Reports вы можете найти по адресу: **http://www.crystalinc.com/crystalreports**.

На заметку В LiveWire Pro имеется поддержка диспетчеров баз данных Informix, Oracle и Sybase, а также поддержка стандарта Microsoft (ODBC). Благодаря ODBC приложение LiveWire Pro может связываться с базами данных, созданными с помощью dBase, Visual FoxPro, и даже с простыми текстовыми файлами.
LiveWire Pro применяет свой интерфейс к библиотекам Informix, Oracle и Sybase с помощью соответствующих API, а не посредством драйверов ODBC. Приложения, разработанные с помощью LiveWire Pro, помогают упростить конфигурацию базы данных и обеспечивают повышенное быстродействие в сравнении с подходом, основанным на использовании ODBC.

LiveWire Pro продается в составе пакета серверов *SuiteSpot* фирмы Netscape. В состав SuiteSpot входят следующие средства:

- сервер предприятия — Web-сервер Netscape;
- сервер каталога — поисковая система, которую можно использовать для создания и поддержания баз данных с использованием всех ресурсов ваших Web-страниц;
- резервный сервер — средство для хранения копий часто используемых файлов на локальной машине;
- почтовый сервер — средство, поддерживающее протокол передачи обычной почты SMTP (Simple Mail Transfer Protocol) для коммуникаций сервер-сервер, а также протокол почтового офиса POP (Post Office Protocol) для связи с клиентами почты;
- LiveWire Pro — инструменты для разработки приложений и средств доступа к базам данных (о котором идет речь в данной главе).

На рис. 36.1 показана структура пакета SuiteSpot.

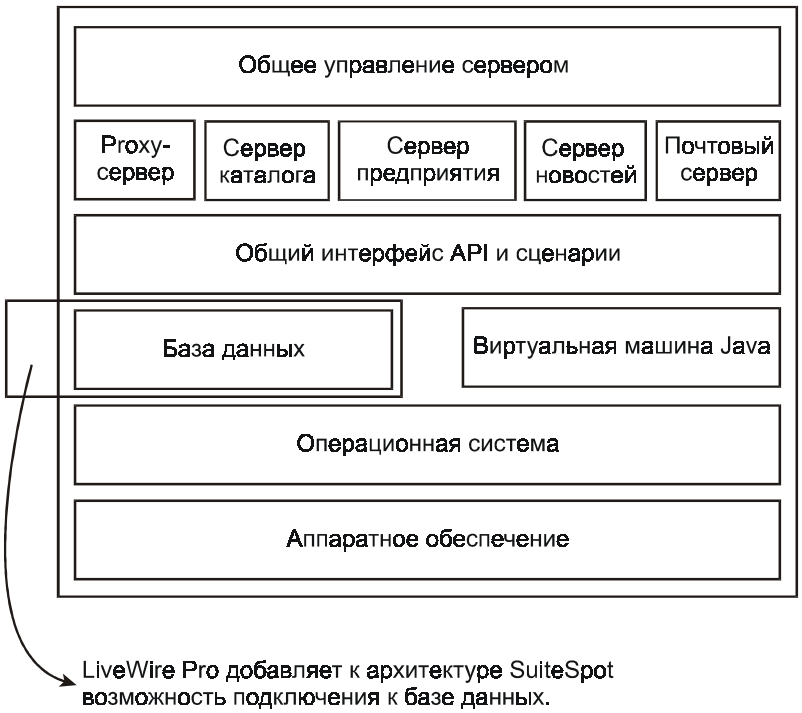

Рис. 36.1. С помощью SuiteSpot можно решить проблему совместимости между различными операционными системами и оборудованием

Стоимость SuiteSpot равна стоимости четырех серверов (сервера предприятия, каталога, резервного и почтового). Если вы решили приобрести эти четыре сервера, купите SuiteSpot и получите в качестве бесплатного дополнения LiveWire Pro. Более подробную информацию вы можете найти на страницах **`http://home.netscape.com/comprod/server_central/product/suite_spot/index.html`**.

Принципы работы LiveWire Pro

Хотя вы вполне можете создавать приложения, не имея представления о том, как работает LiveWire Pro, знание принципов работы этого средства поможет вам в процессе отладки, а также для более эффективного распределения работы между различными компьютерами, имеющимися в вашем распоряжении.

Как известно, пользователи подключаются к Web-странице с помощью *Web-броузеров*, например Netscape Navigator. Эта программа, так называемый *клиент*, запрашивает у Web-сервера определенные объекты, например страницы HTML. Адрес каждого такого объекта называется *универсальный локатор ресурсов* или *URL* (Uniform Resource Locator). Протокол, с помощью которого выполняются запросы и ответы, называется *протоколом передачи гипертекста* или *HTTP* (*H*yper*T*ext *T*ransfer *P*rotocol).

Вероятно, вы знаете также, что кроме статических страниц HTML (которые броузер передает на Web-страницы), можно писать программы, выполняемые на сервере. Эти программы поддерживают протокол CGI, который позволяет получать информацию от пользователя, причем возможности обработки этой информации намного шире возможностей HTTP. Обычно в результате выполнения сценария CGI клиенту возвращается документ HTML и пользователь видит новую страницу.

Краткое знакомство с HTTP

Большинство запросов HTTP служат для пересылки клиенту определенного объекта (обычно это страница HTML) с сервера. В таких запросах содержится ключевое слово `GET`. Если же сервер сконфигурировать определенным образом, то URL могут указывать на программы, которые выполняются (а не отсылаются клиенту) и возвращают результат клиенту. Такие URL соответствуют сценариям CGI, которые можно запустить с помощью метода `GET`.

Другим сценариям CGI требуется дополнительная входная информация, например результат заполнения формы HTML. В таких сценариях используется метод `POST`. Когда сервер распознает запрос `POST`, он запускает сценарий CGI, а затем передает поток информации, поступающий от клиента, в стандартный входной поток (`STDIN`) сценария CGI.

CGI служит для выполнения различных задач. На многих Web-страницах CGI успешно используется для передачи результатов из формы HTML по электронной почте владельцу страницы, для поиска на Web-страницах информации, запрошенной пользователем, или даже для передачи запроса к базе данных. И все же у Netscape имеются причины предлагать продукты в качестве альтернативы CGI.

- Всякий раз при включении программы CGI она запускается, выполняется и завершается. Этот процесс запуска требует большого количества вычислений. Если сценарий CGI занят, то сервер может потратить много времени, многократно повторяя попытки запуска этого сценария.
- Взаимодействие между сервером и сценарием CGI ограничено потоками данных через стандартное устройство `STDIN` и, возможно, несколькими переменными в окружении. Сценарий CGI не может задать серверу вопросы, поэтому сервер должен упаковывать все, что сценарий может запросить, и хранить эту информацию для каждого сценария.
- Сценарии CGI обычно пишутся на языках Perl, Tcl, REXX или даже C и C++ — языках общего назначения, в которых нет встроенных механизмов для работы с протоколом CGI. Многие создатели Web-страниц сталкиваются с определенными трудностями при написании программ на этих языках.
- Сценарии CGI работают непосредственно с операционной системой, поэтому их сложно перенести с сервера UNIX на сервер Windows NT.
- Сценарии CGI могут использоваться для несанкционированного доступа к странице. Хотя имеются средства, позволяющие защитить сценарий CGI от несанкционированного доступа, многие разработчики не знают о них или предпочитают их не использовать. Поэтому некоторые системные администраторы либо не позволяют использовать сценарии CGI на своих машинах, либо настаивают на проверке сценария перед его инсталляцией. Эти ограничения увеличивают расходы и задержки при разработке страницы и могут вообще исключить возможность использования сценариев CGI.
- Сценарии CGI выполняются на сервере, однако многие операции (например, проверка ввода в форму) требуют меньше ресурсов и возвращают результат быстрее, если выполнять их на машине клиента.

Вы можете добавить CGI или его альтернативу для сервера — LiveWire, на сетевой сервер предприятия, а также добавить CGI или LiveWire на сервер Internet. Для сценариев CGI требуется особая конфигурация сервера. Приложение LiveWire следует инсталлировать с помощью диспетчера приложений (Application Manager).

Даже в относительно спокойном окружении корпоративной сети уделите внимание вопросам защиты информации при работе с CGI. Многие сценарии обеспечивают доступ к стратегическим ресурсам предприятия и их необходимо защитить от несанкционированного доступа *изнутри* компании.

Возможности, предлагаемые Netscape

Тем разработчикам Web-страниц, которые хотят выйти за пределы стандартных возможностей, Netscape предлагает два варианта:

- выбор языка программирования для написания приложения;
- выбор системы, на которой будет выполняться это приложение.

Разработчики Web-страниц, которые работают с сервером предприятия, могут использовать приложения, написанные на языке Java (так называемые аплеты). Это объектно-ориентированный язык, разработанный специально для Web фирмой Sun Microsystems. Они также могут писать программы на языке JavaScript — упрощенном языке, основу которого составляет Java. JavaScript предназначен для внедрения в файл HTML и для выполнения на машине клиента. Броузер Netscape понимает язык JavaScript и может выполнять эти программы.

Аплеты Java хранятся на сервере, однако они загружаются на машину клиента и выполняются на ней. Сценарии JavaScript обычно выполняются на компьютере клиента. Если на сервере установлен пакет LiveWire, то аплеты Java можно скомпилировать и выполнять также и на сервере.

На заметку Хотя сценарии JavaScript, обрабатываемые клиентом, и сценарии JavaScript, обрабатываемые сервером (т.е. приложения LiveWire) используют один и тот же язык, LiveWire поддерживает несколько объектов, создаваемых при выполнении на сервере, которые зачастую являются решающими при разработке приложения LiveWire. В разделе "JavaScript на сервере" в этой главе такие объекты будут рассмотрены более подробно.

Программист может также написать приложение для определенной платформы (например, для компьютера Windows или Macintosh) и интегрировать его с броузером Netscape. Эти приложения, называемые модули-приложения (plug-ins), активируются, когда сервер посылает особое сообщение о типе данных MIME, которые может обработать такое приложение (обычно они разрабатываются на языке типа C++).

Созданные ранее вспомогательные приложения открываются в отдельном окне и выполняются как отдельный процесс, а дополнительные модули-приложения интегрированы в приложение-клиент и могут обмениваться сообщениями с броузером Netscape.

На рис. 36.2 показаны некоторые возможности, доступные программисту в окружении Netscape.

На заметку Язык JavaScript ранее назывался "LiveScript". Это название все еще появляется в литературе, а компиляторы и интерпретаторы JavaScript поддерживают его до сих пор. Изменилось только название — язык остался тем же.

Многие разработчики, особенно непрофессиональные программисты, считают, что программирование на языке JavaScript проще, чем программирование на языке Java. С помощью LiveWire можно внедрить в страницу код JavaScript, который выполняется на сервере, после чего результаты работы этого сценария передаются в программу-клиент.

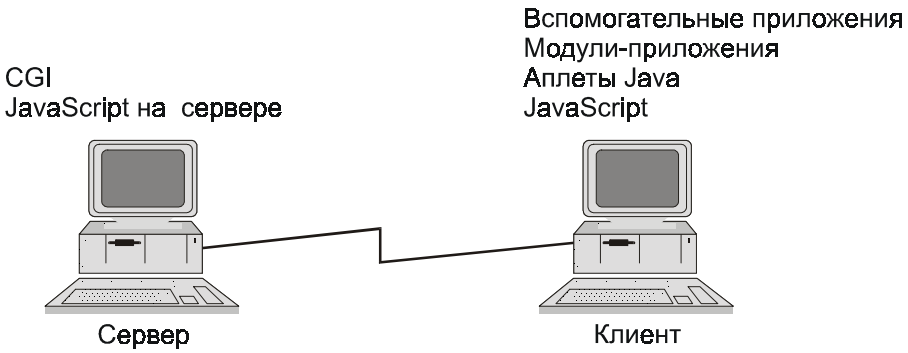

Рис. 36.2. Если Web-страницы работают на серверах Netscape, а пользователь имеет Netscape Navigator, то разработчик может выбрать, где должно выполняться приложение

Как LiveWire обрабатывает запрос

Чтобы понять возможности LiveWire, необходимо иметь представление о том, как LiveWire обрабатывает JavaScript на сервере. В LiveWire Server Extension имеется компилятор сценариев JavaScript. Когда разработчик заканчивает написание страницы, в которой имеется код JavaScript для сервера, этот код передается в компилятор. Компилятор добавляет скомпилированный код в Web-страницу.

Web-сервер обычно обрабатывает запрос `GET`, находя запрашиваемый объект и отсылая его клиенту. Если JavaScript установлен на сервере Netscape, то в этот процесс добавляется еще один шаг. JavaScript регистрирует обращения к определенным URL, поэтому при обращении пользователя к одному из таких URL сервер передает контроль интерпретатору JavaScript в LiveWire Server Extensions. Интерпретатор запускает скомпилированный код, присоединенный к данной странице. Готовый результат, состоящий из статического HTML и динамического вывода программы, возвращается клиенту.

На заметку

В технической документации фирмы Netscape обычно используется термин *live* (*живой*). Netscape использует этот термин как синоним термина *динамический*. Поэтому термины *живой интерактивный документ* и *динамическая Web-страница* означают одно и то же.

Что такое SQL

Как упоминалось выше, единственное различие между LiveWire и LiveWire Pro заключается в том, что LiveWire Pro служит для подключения к реляционным базам данных. В этом разделе речь пойдет о системах управления реляционными базами данных (СУРБД) и о языке SQL, который используется в этих системах.

Некоторые авторы Web-страниц, более уверенно чувствуют себя, работая с такими СУБД, как dBase, чем с новыми программами типа Visual FoxPro или Microsoft SQL Server. Многие современные или более мощные программы используют SQL. Язык SQL возник на ранней стадии развития СУРБД и вышел победителем среди подобных продуктов. Нереляционные базы данных, такие как объектно-ориентированная база данных Object Design, ObjectStore, часто предлагают интерфейс SQL в дополнение к их собственному языку работы с данными.

На заметку

SQL начал свое существование как язык IBM, но в 1988 году Американский институт национальных стандартов (ANSI) и Международная организация по стандартизации (ISO) утвердили стандарт этого языка ISO-ANSI SQL. Стандарт ISO-ANSI 1988 года описывает хорошо разработанный язык, но коммерческие приложения зачастую не полностью соответствуют стандарту. Например, в стандарт 1988 года не вошел механизм для создания индексов — средство, которое требуется для всех коммерческих применений.

Редакция стандарта ISO-ANSI 1989 года была более полной, однако все же недостаточно подробной для фирм-разработчиков. Netscape рекомендует разработчикам LiveWire Pro использовать формат запроса из стандарта 1989 года. В большинстве коммерческих программных продуктов теперь поддерживается стандарт 1989 года.

Стандарт ANSI 1992 года гораздо более полный, чем предыдущие редакции. Достаточно сказать, что эта редакция стандарта имеет объем в четыре раза больший, чем редакция 1989 года. Национальный институт стандартов и технологии США (NIST) сертифицировал большинство фирм-продавцов баз данных на соответствие исходному стандарту ANSI 92.

Реляционная модель данных

В большинстве профессиональных СУБД используется реляционная модель данных. Реляционная модель характеризуется наличием одного или нескольких "отношений" (relations), которые обычно называют *таблицами* (рис. 36.3). LiveWire обеспечивает непосредственный доступ к таблице с помощью библиотеки подключения к базе данных (Database Connectivity Library).

В хорошо спроектированной базе данных каждая таблица описывает определенное понятие. Рассмотрим в качестве примера модель понятия "книга" для базы данных оптового продавца книг. В каждой строке таблицы `Книга` содержится одна запись — информация об одной книге. Столбцы представляют поля записи — сведения о книге, которые должны храниться в приложении, например название, год издания и розничная цена. В каждой таблице должно быть определенное сочетание столбцов (обычно это один столбец), которое уникальным образом идентифицирует каждую строку. Этот набор столбцов называется *первичным ключом* (*primary key*). Для таблицы `Книга` таким ключом может быть классификация ISBN.

В таблице могут также содержаться указатели на другие таблицы, так называемые *внешние ключи* (*foreign keys*), они представляют собой основные ключи из другой таблицы. Например, на рис. 36.4 каждая книга ассоциируется с издательством посредством ключевого поля издательства в записи о книге. В таблице `Книга` идентификационный код издательства является внешним ключом. В таблице `Издательство` этот идентификационный код издательства является первичным ключом.

ISBN	Title	Publication Year	Retail Price	Publisher ID
0-7897-0801-9	Webmaster Expert Solutions	1996	59.99	7897
1-57521-070-3	Creating Web Applets with Java	1996	39.99	57521
0-7897-0790-X	Enhancing Webscape Web Pages	1996	34.99	7897
1-56205-473-2	WebmastersX Professional Reference	1996	55.00	56205
1-57576-354-0	An Interactive Guide to the Internet	1996	75.00	57576
1-57521-016-9	Bots & Other Internet Beasties	1996	49.99	57521
1-56205-573-9	Building Internet Database Servers/CGI	1996	45.00	56205
1-57521-049-5	Java Unleashed	1996	49.99	57521
0-7897-0758-6	Special Edition Using HTML, Second Edition	1996	49.99	7897
0-7897-0604-0	Special Edition Using Java	1996	49.99	7897
1-57521-073-8	Teach Yourself JavaScript in a Week	1996	39.99	57521
0-7897-0753-5	The Big Basic Book of the Internet	1996	19.99	7897
1-56205-521-6	Flying Through the Web: VRML	1996	30.00	56205

Рис. 36.3. Каждая таблица определяется столбцами и ключами. Данные в таблице располагаются в строках

Разработке баз данных посвящен отдельный раздел компьютерной науки. Если вам нужно спроектировать новую базу данных, но вы не имеете соответствующего опыта, то лучше обратитесь за помощью к специалисту. При разработке реляционной базы данных следует всегда находить компромисс между двумя противоположными тенденциями. Если таблицы имеют слишком много столбцов (полей), то становится сложно поддерживать согласованность данных. Например, если в таблицу `Книги` включить не только код издательства, но также и его адрес, то при изменении адреса в таблице `Издательство` он останется без изменения в таблице `Книга`.

Чтобы обеспечить согласованность, необходимо разделить базу данных на несколько таблиц. Однако, если в большой базе данных имеется слишком много маленьких таблиц, то понадобится слишком много запросов для поиска информации в этих таблицах с учетом внешних ключей. Большие базы данных, в которых отдельные таблицы имеют слишком мало информации, могут занимать много места на диске и работать очень медленно.

В теории баз данных используется понятие пяти уровней *нормализации*, то есть стандартов, необходимых для поддержания согласованности базы данных. Нормальные формы являются иерархическими. База данных в третьей нормальной форме удовлетворяет требованиям для первой, второй и третьей нормальных форм. Вот определения пяти нормальных форм.

- **Первая нормальная форма.** Каждое пересечение строки и столбца имеет единственное значение. Тогда база данных, в которой все книги, опубликованные данным издательством в 1996 году, хранятся в одном пересечении строки и столбца, нарушает определение первой нормальной формы.
- **Вторая нормальная форма.** Каждый неключевой столбец должен зависеть от основного ключа. Если основной ключ является составным (имеет более одного компонента), то неключевой столбец может быть подмножеством основного ключа. На практике для второй нормальной формы требуется, чтобы основной ключ состоял только из одного столбца.

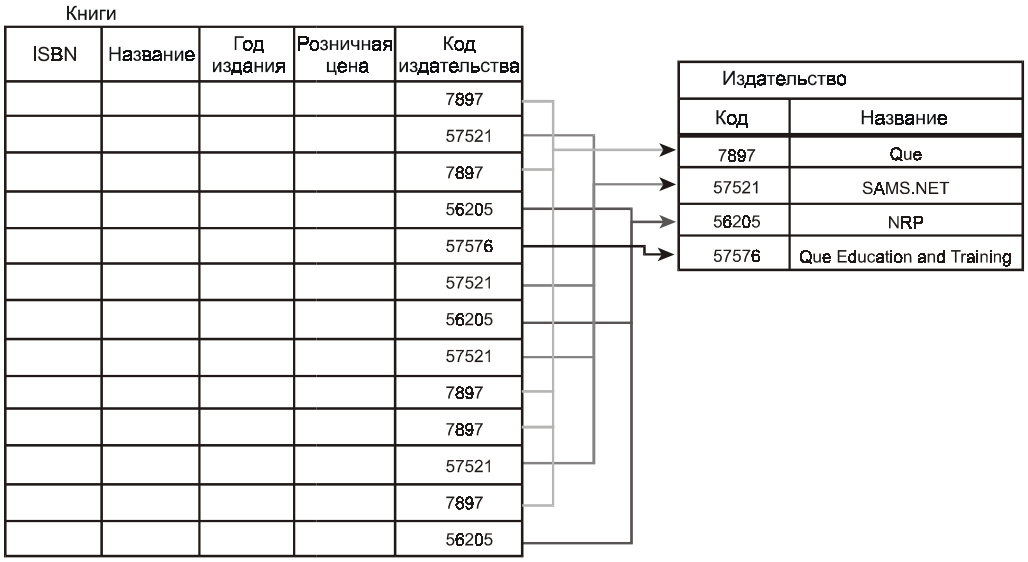

Рис. 36.4. Связи между таблицами устанавливаются посредством внешних ключей

Например, код ISBN является уникальным для каждой книги (он указывается в выходных данных книги). В этом коде имеется несколько внутренних полей, в том числе поле для издательства и поле для названия книги. Если в таблице код ISBN представлен в качестве составного первичного ключа (например, в одном столбце содержатся идентификационные коды издательств, а в другом — идентификационные коды книг), и в этой же таблице имеется столбец `Адрес издательства`, который зависит *только* от идентификационного кода издательства, то такая таблица будет нарушать вторую нормальную форму.

- **Третья нормальная форма.** Неключевой столбец может зависеть от другого неключевого столбца. Каждый столбец должен соотноситься с определенным основным ключом.
- **Четвертая нормальная форма.** Независимые отношения один-ко-многим между столбцами основного ключа и неключевыми столбцами не допускаются. Например, в табл. 36.1 нарушена четвертая нормальная форма, поскольку поля `Посещенные города` и `Дети` не зависят одно от другого. Автору, у которого нет детей и который не посетил ни одного города, будет соответствовать пустая строка.
- **Пятая нормальная форма.** Разбиение таблиц на наименьшие возможные составные части во избежание избыточной информации в таблице. В предельном случае таблицы в пятой нормальной форме могут состоять из основных ключей и одного неключевого столбца.

Таблица 36.1. Для таблицы, которая не соответствует четвертой нормальной форме, характерны многочисленные пустые ячейки

Автор	Дети	Посещенные города
Бреди	Грег	Сиэтл
Бреди	Синди	Лос-Анджелес
Бреди	Бобби	
Клинтон	Челси	Вашингтон
Клинтон		Лос-Анджелес
Клинтон		Сент Луис

Базы данных не следует без разбора приводить к пятой нормальной форме. Такие базы данных часто имеют высокую степень интеграции, но могут занимать слишком много места на диске (поскольку таблицы имеют много внешних ключей). Кроме того, вполне вероятно, что таблицы в пятой нормальной форме будут медленно работать, поскольку даже для простого запроса потребуется провести поиск во множестве таблиц. Всегда стремитесь при разработке баз данных к оптимальному сочетанию согласованности и быстродействия.

Пустое пересечение столбца и строки называется *null*. В спецификации для каждой таблицы должно быть указано, в каких столбцах допускаются пустые ячейки.

Общие сведения о SQL

Типичный цикл работы базы данных примерно таков.

1. База данных создается с помощью команды SQL `CREATE DATABASE`:

```
CREATE DATABASE bookWholesale
```

2. Таблицы создаются с помощью команды `CREATE TABLE`:

```
CREATE TABLE books
(isbn char(10) not null,
title char(20) not null,
publicationYear datetime null,
retailPrice money null))
```

3. Создается один или несколько индексов:

```
CREATE INDEX booksByYear ON books (publicationYear)
```

Многие СУРБД поддерживают *групповые* индексы. В групповом индексе данные физически хранятся на диске отсортированными по индексу. Групповой индекс несколько усложняет работу базы данных при добавлении или удалении элементов, но благодаря ему можно достичь очень высокого быстродействия, если число читаемых записей намного больше по сравнению с количеством обновлений. В каждой таблице можно использовать не более одного группового индекса.

SQL поддерживает ключевое слово `UNIQUE`, которое означает, что две строки в СУРБД не могут иметь одинаковый индекс.

4. Таблицы заполняются данными:

```
INSERT INTO books VALUES ('0789708019', 'Webmasters Expert Solu-
tions', 1996, 69.95)
```

В некоторых базах данных новые строки добавляются часто, а в других — однажды составленная база данных надолго остается неизменной.

5. Выполняются запросы к базе данных:

```
SELECT title, publicationYear WHERE retailPrice < 40.00
```

В большинстве случаев базы данных создаются именно для того, чтобы с помощью запросов облегчить поиск нужной информации.

6. Данные можно изменять:

```
UPDATE books
SET retailPrice = 59.95
WHERE ISBN='0789708019'
```

7. Данные можно удалять из таблицы:

```
DELETE FROM books
WHERE publicationyear < 1990
```

8. Как отдельные таблицы, так и всю базу данных можно удалить, если они вам больше не нужны:

```
DROP TABLE books
DROP DATABASE bookWholesale
```

Если количество запросов велико по сравнению с количеством добавлений, удалений и обновлений записей, индексы могут повысить быстродействие. Если частота изменения базы данных повышена, изобилие индексов будет затруднять работу приложения.

Когда таблица создана, нужно указать тип данных для каждого столбца. Все СУРБД поддерживают тип character (знак) и integer (целое число). Большинство коммерческих СУРБД также поддерживают другие типы данных: числа с плавающей точкой (действительные), денежные единицы, дата, время и даже специальные двоичные типы данных для хранения звуков, изображений и других объектов.

Database Connectivity Library в LiveWire помогает совместить нейтральный набор типов данных с типами данных, которые поддерживаются определенной СУРБД.

Что такое транзакции

Во многих приложениях пользователю необходимо сгруппировать несколько команд, чтобы они работали согласованно, как одно целое. Такое целое называется транзакцией. На следующем примере поясняется, для чего необходимы транзакции.

1. Вы связываетесь с Агентом №1, чтобы заказать билет на Гавайи. Агент №1 открывает базу данных и считывает из нее свободные места. Он находит, что на нужном вам рейсе осталось только одно место.
2. Пока вы решаете, взять ли этот билет, другой пассажир обращается к Агенту №2, чтобы заказать билет на тот же рейс. Агент №2 запрашивает базу данных, узнает, что имеется одно свободное место, и сообщает об этом своему клиенту.
3. И вы, и другой пассажир решили взять билет. Оба агента вводят данные одновременно. Агент №1 оказался немного проворнее. Он изменил базу данных и сообщил вам о принятии вашего заказа. Затем обрабатывается заказ, сделанный Агентом №2, и после изменения базы данных Агентом №2 получается, что билет продан другому пассажиру.
4. В аэропорту выясняется, что вас нет в списках на резервирование места в самолете, и он улетает на Гавайи без вас, а вы вспоминаете всех предков компьютера, агента и даже аэропорта, волоча свои чемоданы домой...

Этот пример иллюстрирует классическую проблему базы данных — *потерянное обновление*. Квалифицированный программист SQL смог бы решить проблему утерянных обновлений, начав транзакцию перед обработкой запроса. База данных дает агенту по продаже билетов возможность чтения данных, но при этом он не сможет обновить базу данных. Когда билетный агент начинает вводить в базу данных информацию о продаже билета, приложение запрашивает для него исключительное право на запись. Пока у агента имеется исключительное право на запись, никто другой не сможет прочитать или записать те же данные. После того как агент получит исключительное право на запись, приложение запросит базу данных, свободно ли еще данное место. Если место еще свободно, приложение обновит базу данных, отметив это место как проданное. После этого транзакция закончится, и в базу данных будут внесены изменения. Вот как выглядит сценарий потерянных обновлений с использованием транзакции.

1. Вы связываетесь с Агентом №1, чтобы заказать авиабилет на Гавайи. Агент №1 открывает базу данных и узнает, что имеется только одно свободное место на нужный вам рейс.
2. Пока вы раздумываете, брать ли билет, другой пассажир связывается с Агентом №2 с заказом на тот же самый рейс. Агент №2 запрашивает базу данных и узнает об одном свободном месте, о чем сообщает своему клиенту.
3. И вы, и другой пассажир решили взять билет. Оба агента ввели данные одновременно. Агент №1 был немного проворнее — он запустил транзакцию и изменил базу данных. Когда транзакция завершилась, агент подтвердил, что ваш заказ принят. Примерно в это же время в базу данных поступает информация о резервировании билета от Агента №2. Когда Агент №2 пытается получить доступ к транзакции разрешения записи, он получает сообщение о том, что база данных в данный момент изменяется. Когда ваша операция завершилась, агент обнаруживает, что он не может продать билет, поскольку он уже был куплен. Агент продолжает работать с пассажиром, чтобы подобрать для него другой рейс.
4. Вы прибыли в аэропорт. Ваше имя имеется в компьютере, и заказ подтвержден. Наслаждайтесь отдыхом!

Транзакции используются также для восстановления системы. Поскольку запись на жесткий диск, часто через сеть, занимает довольно много времени, то многие базы данных некоторое время хранят обновления в локальных буферах. Если в системе произойдет сбой прежде, чем СУРБД действительно обновит базу данных, то некоторые из этих обновлений могут быть утеряны. В большинстве коммерческих продуктов информация о каждом изменении базы данных записывается на жестком диске в специально отведенном для этого месте, которое называется *журнал транзакций* (*transaction log*). Если сбой происходит прежде, чем обновления были внесены в базу данных, то последовательность транзакций может быть воспроизведена в процессе восстановления системы, и обновление будет отражено в базе данных.

Что такое курсор

Все, кому приходилось работать с базами данных, ориентированными на PC, привыкли к тому, что запросы возвращают только одну запись. Например, в dBase III имеется концепция *указателей*. Программист может написать:

```
GOTO 3
DISPLAY
```

и dBase вернет все поля третьей записи. Затем программист может ввести команду

```
DISPLAY NEXT 1
```

и программа, передвинув указатель на одну запись, покажет четвертую по счету запись.

Многие программисты SQL считают, что указание только на одну запись является ужасно неудобным. В SQL можно использовать такую команду:

```
SELECT * WHERE publicationYear = 1996
```

Этот запрос может вернуть нуль, одну или несколько записей. В некоторых случаях программист уверен, что должна быть возвращена именно одна запись, например, в случае такого обращения к индексному полю:

```
SELECT * WHERE ISBN='0789708019'
```

Однако сущность языка SQL такова, что программа все-таки будет работать так, как если бы речь шла о наборе записей.

Многие коммерческие приложения SQL поддерживают понятие *курсора*. Курсор похож на указатель в dBase — он указывает на одну запись в каждый момент времени и может быть передвинут вперед или назад в наборе записей. LiveWire Pro поддерживает структуру возврата данных, основанную на курсоре. Чтобы настроить курсор, нужно указать

```
myCursor = database.cursor (selectStatement, updateFlag);
```

где *selectStatement* соответствует объявлению SELECT в SQL стандарта ANSI-89, а *updateFlag* (которое может иметь значения TRUE или FALSE) управляет тем, может ли определенный курсор обновлять базу данных.

На заметку

В объектно-ориентированном языке C++ методы объекта определяются с помощью разделителя-точки. Если программист создал новый объект theAircraft (самолет) и хочет, чтобы он поднялся на высоту 10000 футов, то он может написать `theAircraft.climb(10000);` Однако в C++ для этой цели чаще используется переменная, в которой сохраняется адрес объекта. Такая переменная называется *указатель* (это название не имеет ничего общего с указателем в dBase). Чтобы вызвать метод объекта с помощью указателя, программист использует запись со стрелкой, примерно так: `theAircraftPointer->climb(10000);`

Указатели (в смысле C и C++) являются мощным средством, однако непосредственный доступ к ячейкам памяти является рискованным в смысле защиты информации. В отличие от C++, языки Java и 3JavaScript создают не указатели на объекты, а новые объекты, при этом программист использует запись с точкой вместо записи со стрелкой.

После создания курсора программист может перемещать его по строкам, которые выбираются с помощью команды `SELECT`. Например, команда

```
myCursor.next()
```

перемещает курсор на новую выбранную строку.

Основные сведения о Crystal Reports

Многим разработчикам Web-страниц ежедневно приходится тратить много времени на создание запросов SQL. Однако если вы запускаете LiveWire на сервере Windows NT, то можете использовать для подготовки запросов программу Crystal Reports, входящую в комплект LiveWire Pro. Использование Crystal Reports дает следующие преимущества.

- *Подробные отчеты по разделам и промежуточные отчеты.* Отчет может состоять из нескольких разделов. Кроме того, разработчик может написать отдельный отчет, а затем встроить его в качестве составной части в основной документ.
- *Условные отчеты.* Разделы подробного отчета и текстовые объекты можно изменять в зависимости от данных. Например, запись может иметь переключатель языка, что позволяет пользователю напечатать отчет на английском или испанском языках.
- *Распространение отчетов в Internet.* Отчеты можно распространять в Internet с помощью экспортирования отчета в HTML.
- *Отчеты в виде форм.* Текст и объекты можно разместить на странице с помощью сетки, направляющих и линеек.
- *Отчеты в виде таблиц.* Отчеты могут представлять итоговую информацию в выразительном двумерном формате.

В последней версии Crystal Reports имеется приложение Report Designer, с помощью которого пользователь может поместить на странице поля, текст и другие элементы в качестве графических объектов.

Database Connectivity Library

В разделе “Общие сведения о SQL” этой главы был рассмотрен типичный жизненный цикл базы данных. Большинство Web-страниц, интегрированных с базами данных, позволяют посетителям посылать запрос к базе данных, а иногда даже добавлять или удалять данные. Иногда вам придется добавлять или удалять таблицы, индексы или даже целые базы данных.

В тех случаях, когда встроенный API является недостаточно мощным средством для обработки приложения, программист может использовать *проходящий SQL* (*passthrough SQL*) — механизм для пересылки SQL в нужную базу данных. Например, программист может использовать такой код:

```
database.execute ("CREATE TABLE books
    (isbn char(10) not null,
    title char(20) not null,
    publicationYear datetime null,
    retailPrice money null)");
```

ВНИМАНИЕ!

Как показывает название, проходящий SQL не интерпретирует код SQL, а передает его непосредственно в целевую СУРБД. При этом код может различаться в зависимости от установленной базы данных. Проходящий SQL используется для создания новых баз данных, однако он не может игнорировать механизм курсора и возвращать строки в качестве набора. При выборе данных должен использоваться встроенный механизм курсора, а не обычный вызов с помощью SQL.

Как открыть и закрыть подключение

Вспомните, что сценарии CGI запускаются для каждого запроса HTTP. Этот процесс требует очень большого количества работы. В отличие от сценариев CGI, приложения LiveWire продолжают выполняться до тех пор, пока вы их явно не выключите. Одно из дополнительных преимуществ такого подхода заключается в том, что приложение LiveWire Pro может открыть подключение к базе данных при запуске и оставить это подключение открытым практически навсегда.

Обычно при запуске приложения LiveWire Pro оно сразу же подключается к базе данных. Для этого используется следующий синтаксис:

```
database.connect(тип БД, имя сервера, имя пользователя, пароль, имя БД);
```

где *тип БД* это одна из следующих баз данных:

- ORACLE
- SYBASE
- INFORMIX
- ILLUSTRA
- ODBC

а *имя сервера*, *имя пользователя*, *пароль* и *имя БД* — это обычная информация, необходимая для доступа к базе данных.

Другие запросы к этому приложению (от того же самого клиента, но для другой страницы, или от других клиентов) используют одно и то же подключение к базе данных. На рис. 36.5 показано несколько приложений и клиентов, которые взаимодействуют с базой данных. Быстродействие при последовательных запросах к приложению повышается, поскольку для выполнения каждого запроса не нужно выполнять повторный запуск приложения.

Приложение может проверить наличие подключения с помощью метода `connected()`. В следующих строках кода показано, как выполнить подключение и проверить, найдена ли база данных и успешно ли выполнено подключение.

```
database.connect (INFORMIX, theServer, mmorgan, mySecretWord, de-
moDB);
if (!database.connected())
    write("Ошибка подключения к базе данных.");
else
.
.
.
```

Система хранит информацию о подключениях приложений к базам данных на сервере в памяти общего пользования. Со временем подключение распространяется на другие копии процесса Netscape Server. На рис. 36.6 показан этот механизм, который называется *диффузия*. Программист в любое время может отключить приложение от базы данных — это приведет к тому, что все копии сервера будут отключены от этой базы данных. Такое отключение может понадобиться по двум причинам.

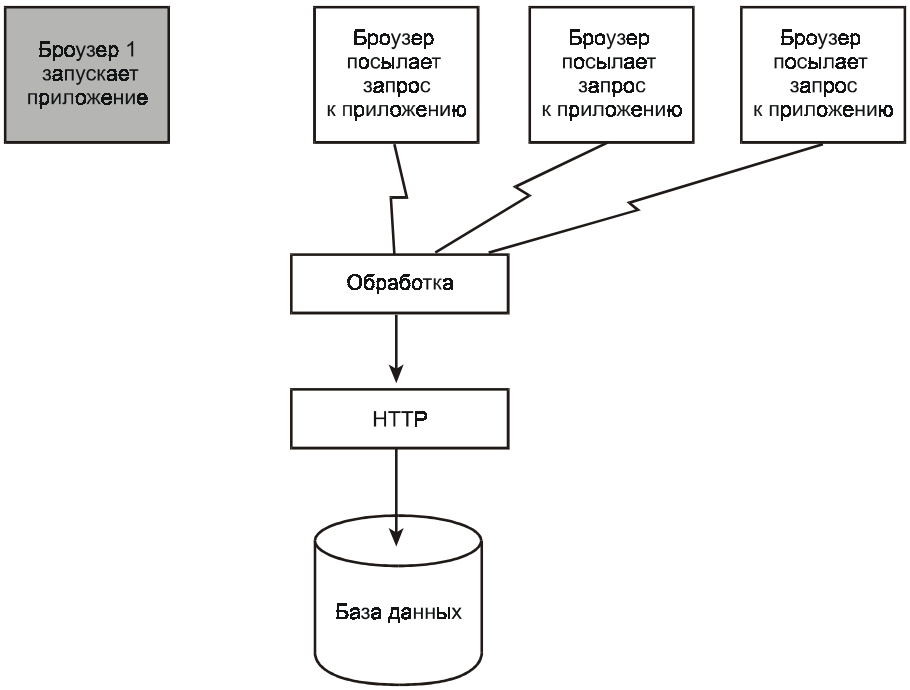

Рис. 36.5. Когда система достигнет устойчивого состояния, вам не придется тратить время на запуск или подключение приложения к базе данных

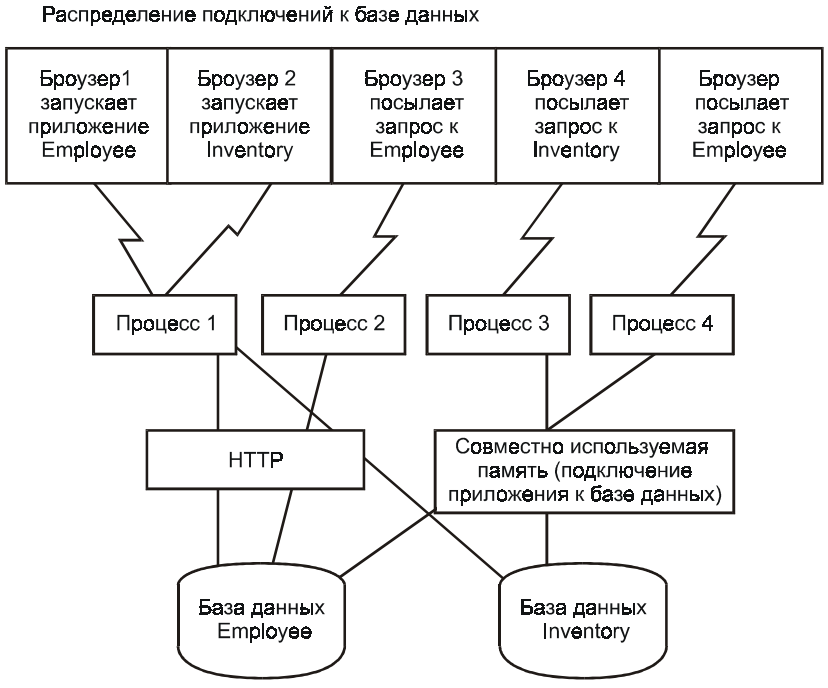

Рис. 36.6. Подключения к базе данных распределяются на сервере

- Приложение может иметь только одно открытое подключение в каждый момент времени, и вам нужно переключить это приложение на другую базу данных.
- В СУРБД обычно ограничено максимальное количество подключений. Отключение приложения, которое больше не нуждается в соединении, высвобождает это подключение для использования другим приложением

Какой бы ни была причина отключения приложения от базы данных, это легко сделать. с помощью такого кода:

```
database.disconnect();
```

СУРБД Informix, связанная с LiveWire Pro, допускает только одно подключение. Хотя эта СУРБД подходит для разработки, большинство разработчиков Web-страниц предпочитает использовать базу данных с большим количеством подключений для использования в интерактивном режиме.

Добавление данных в таблицы

Все обновления должны выполняться посредством обновляемых курсоров. Ниже приводится фрагмент кода JavaScript, который создает новый обновляемый курсор и добавляет новую строку.

```
myCursor = database.cursor("SELECT isbn, title, publicationYear, re-
tailPrice FROM books", TRUE);
myCursor.isbn="078970255x9";
myCursor.title = "Running a perfect Netscape Site";
myCursor.publicationYear = 1996;
myCursor.retailPrice = 49.99;
myCursor.insertRow (books);
```

Удаление строк

Удалить строки легко. Начните с обновляемого курсора и укажите на строку, которую необходимо удалить. Теперь вызовите метод курсора `deleteRow`. Например, чтобы удалить строку, которая соответствует названию полностью распроданной книги, можно написать так:

```
myCursor =database.cursor ("SELECT * FROM books WHERE
    isbn = request.discontinuedBookISBN", TRUE);
myCursor.deleteRow(books);
```

Доступ к данным в отдельной строке

С помощью курсоров LiveWire Pro можно организовать доступ к данным в отдельной строке и получить значение, которое хранится в ячейке, находящейся на пересечении строки и столбца. Допустим, что в базе данных `bookWholesale` имеется таблица под названием `books`, а в этой таблице — столбец `retailPrice`. Чтобы курсор указывал на определенную строку в этой таблице, можно написать так:

```
thePrice = myCursor.retailPrice;
```

Кроме того, можно установить курсор, неявно указав порядок сортировки:

```
myCursor = database.cursor(SELECT MAX(retailPrice) FROM books);
mostExpensiveBook = myCursor[0];
```

Имена столбцов в списке SELECT можно заменить их порядковыми номерами, например:

```
myCursor = database.cursor(SELECT *FROM books);
firstColumnName = myCursor.columnName(0);
secondColumnName = myCursor.columnName(1);
```

Для вставки или удаления записей, а также для изменения полей записи можно использовать обновляемый курсор. Например, чтобы изменить цену книги в таблице books, можно использовать такой код:

```
myCursor = database.cursor("SELECT * FROM books WHERE isbn =
'0789708019',updatable);
myCursor.retailPrice = 59.95;
myCursor.updateRow(books);
```

Доступ к набору данных

Иногда требуется показать все данные из таблицы в виде списка. Для этого можно создать курсор и показать с помощью цикла все строки, выбранные из таблицы. Кроме того, в LiveWire Pro можно использовать функцию SQLTable.

Если вызвать функцию

```
database.SQLTable(selectStatement);
```

то Database Connectivity Library отобразит в таблице HTML данные, отобранные с помощью оператора SELECT, вместе с заголовками столбцов.

В приложении можно вызвать список записей, как показано на рис. 36.7, где каждая запись имеет гиперсвязь со страницей, содержащей более подробную информацию об одной записи. Пример такой страницы вы видите на рис. 36.8. Более эффективный способ выполнения этой задачи — создание на странице со списком курсора для выбора подходящих записей и форматирование полей в HTML. В страницу с одной записью передается основной ключ, затем на основании этого ключа создается курсор. С помощью объявления SELECT все поля записи, ассоциированной с этим ключом, выводятся на экран.

Использование крупных бинарных объектов (BLOb)

В контекстно-ориентированных приложениях, типичных для Web, часто требуется хранить в базе данных изображения, программные продукты, аудио- или видеоклипы. Для решения этой задачи в SQL был добавлен новый тип базы данных, который называется *Крупный бинарный объект* (*Binary Large Object*, или *BLOb*). Например, оптовый продавец книг может хранить в такой базе данных изображения обложек книг. Общий синтаксис для выбора изображения из BLOb и вывода его на экран с помощью дескриптора изображения HTML имеет такой вид:

```
myCursor.blobFieldName.blobImage (imageFormat, ALTstring, ALIGN-
string, ISMAP);
```

Поля ALTstring, ALIGNstring и ISMAP являются необязательными. Если они имеются, то их используют в дескрипторе изображения HTML. Таким образом, в приложении bookWholesale можно написать:

```
myCursor.cover.blobImage("gif", "Обложка книги", "Left", ISMAP);
```

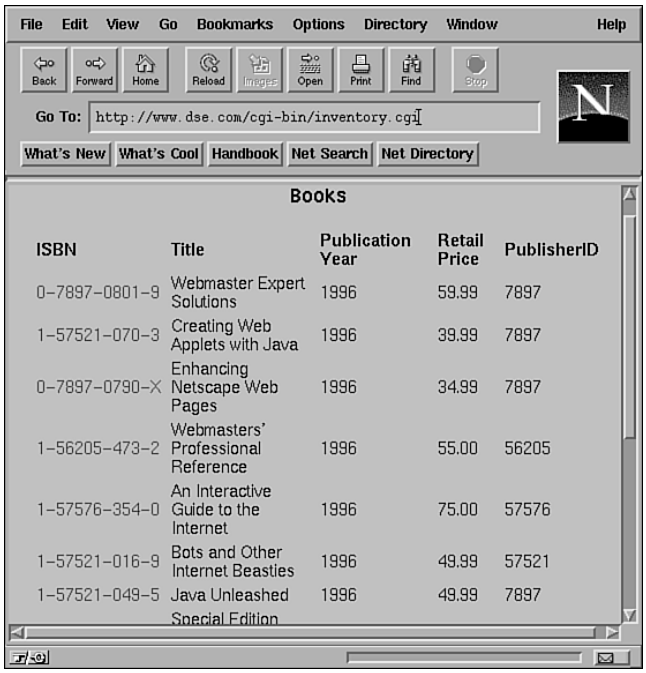

Рис. 36.7. Пользователь может выбрать запись из списка и просмотреть более подробный вариант этой записи

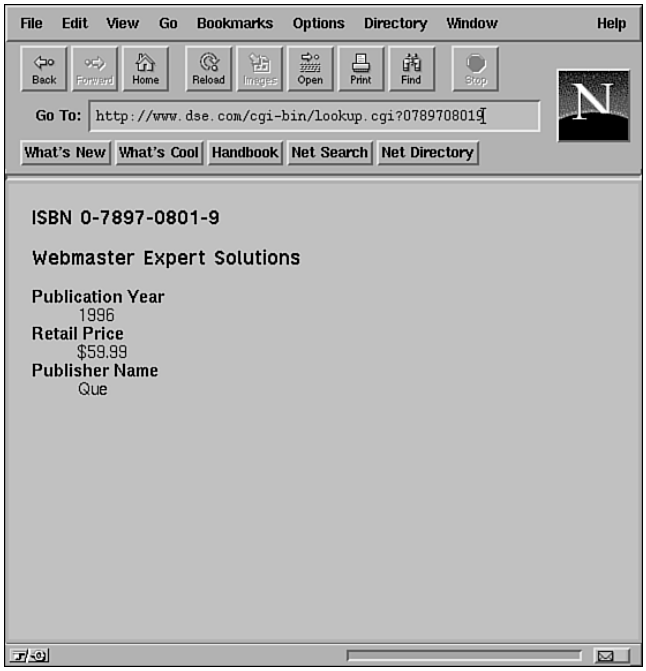

Рис. 36.8. После выбора записи из списка пользователь увидит на экране страницу с одной записью, причем на этой странице содержатся более подробные данные

В базе данных BLOb можно использовать гиперссылки, что дает возможность читать содержимое базы данных с помощью вспомогательных приложений, например:

```
blobFieldName.blobLink(mimeType, linkText);
```

Такая конструкция обычно используется с большими BLOb, например, с аудиоклипами. Сервер Netscape хранит BLOb в памяти до тех пор, пока пользователь не щелкнет на другой ссылке, или до истечения 60 секунд — какое бы из этих событий ни произошло первым. Вот пример того, как переслать BLOb клиенту:

```
myCursor = database.cursor ("SELECT * FROM blobbedBooks");
while (myCursor.next())
{
    write (myCursor.isbn);
    write (myCursor.cover.blobImage("gif"));
    write (myCursor.authorReading.blobLink("audio/x-wav",
           "Избранные фрагменты из" + myCursor.title);
    write ("<BR>");
}
```

Этот код показывает GIF-изображение обложки книги. Если пользователь выбрал эту ссылку, клиент загружает и проигрывает в течение нескольких секунд аудиоклип — имя автора и отдельные фрагменты из данной книги.

BLOb можно добавить в записи примерно таким же образом, как и другие данные. Например:

```
myCursor = database.cursor("SELECT * FROM blobbedBooks, TRUE);
myCursor.isbn="X0789708019";
myCursor.cover = blob("CoverOfWebmasters.gif");
myCursor.insertRow("blobbedBooks");
```

Транзакции LiveWire Pro

Управление транзакциями выполняется с помощью трех методов базы данных:

- `beginTransaction()` — начать транзакцию;
- `commitTransaction()` — выполнить транзакцию;
- `rollbackTransaction()` — отменить транзакцию.

Эти методы можно использовать для создания примерно такого кода:

```
database.BeginTransaction();
int db_error = 0;
dbError = database.execute("INSERT INTO books(isbn, title)
            VALUES (request.isbn, request title);
if (!dbError)
{
    dbError = database.execute ("INSERT INTO authors
        VALUES (request.isbn, request.author1));
    if(dbError)
        database.rollbackTransaction();
    else
        database.commitTransaction();
}
else
//При обработке произошла ошибка
database.rollbackTransaction();
```

Обработка ошибок

LiveWire Pro позволяет СУРБД работать достаточно независимо, без вмешательства программиста. Однако если имеет место сбой, то программисту нужно более подробное сообщение об ошибке — такие сообщения можно создать с помощью СУРБД. В подобных случаях Database Connectivity Library возвращает два типа сообщения об ошибке.

Каждый вызов API возвращает код ошибки. Программист может проверить, имеется ли ошибка. Если нет (`FALSE`), то ошибка не обрабатывается. Если же произошла ошибка (`TRUE`), то программа возвращает коды, соответствующие типу ошибки (например, ошибка сервера, ошибка библиотеки, подключение недоступно, недостаточно памяти).

Если ошибка связана с сервером или библиотекой, то с помощью четырех функций, перечисленных ниже, можно получить более подробную информацию об ошибке:

- `database.majorErrorCode` — возвращает код ошибки SQL;
- `database.majorErrorMessage` — возвращает словесное описание, соответствующее основному коду ошибки.
- `database.minorErrorCode` — возвращает дополнительный код, посланный библиотекой базы данных, например степень опасности ошибки.
- `database.minorErrorMessage` — возвращает дополнительное сообщение, которое передается из библиотеки базы данных.

Если запустить утилиту JavaScript `trace`, то можно увидеть на экране все коды ошибок и соответствующие сообщения.

Серверы JavaScript и серверы Netscape второго поколения

Языки Java и JavaScript играют основную роль в новых серверах FastTrack и Enterprise, а также в серверах почты, новостей, каталога и резервном. Каждый сервер использует виртуальную машину Java и понимает JavaScript. Более того, каждый сервер имеет привязки к Database Connectivity Library. Следовательно, программист может указать серверу, что в базе данных нужно хранить информацию о самом сервере и о его работе, а затем можно передать эту информацию в сеть посредством LiveWire Pro.

Языки Java и JavaScript

Java — это объектно-ориентированный язык программирования, ориентированный на Web. Как и в традиционных языках, например C и C++, перед выполнением программы код Java должен быть скомпилирован. При выполнении программы создаются объекты, основанные на их описании, составленном программистом или взятом из библиотек класса данного языка.

В отличие от традиционных языков, Java не компилируется в исходный набор инструкций конкретного компьютера — он компилируется в коды, которые не зависят от оборудования. В продуктах Netscape (например, в Netscape Navigator) имеется интерпретатор этих кодов.

Разработчики страниц HTML могут внедрять в свои страницы приложения Java (так называемые аплеты). Когда пользователь работает со страницей, аплет загружается, выполняется и выполняет определенные операции на сервере.

JavaScript — это интерпретируемый язык, родственный языку Java. Программы JavaScript хранятся в исходном виде в странице HTML. В сеансе работы страница и имеющийся в ней код JavaScript загружается клиентом Netscape, затем JavaScript интерпретируется и выполняется.

JavaScript на сервере

Если пакет LiveWire установлен на сервере, то можно вызвать компилятор LiveWire как:

```
lwcomp [-cvd] -o binaryFile file
```

где *binaryFile* — имя выходного файла (который обычно имеет расширение `WEB`), а *file* — имя входного файла. Если входной файл состоит из сочетания HTML и JavaScript, то он имеет расширение `HTML` (или `HTM` в среде DOS/Windows). Если входной файл написан на JavaScript, то ему присваивается расширение `JS`.

В табл. 36.2 показаны пять ключей командной строки, которые используются при компиляции LiveWire.

Таблица 36.2. Опции командной строки компилятора

Ключ	Значение
`-c`	Только проверка, двоичный файл не создается
`-v`	Подробная информация о процессе компиляции
`-d`	Компиляция с отладочной информацией
`-o binaryFile`	Выходному файлу присваивается это имя
`-h`	Справка — показывает справочное информационное сообщение

Опция `-v` дает так много полезной информации, что ее почти всегда стоит включать. Рекомендуем вам всегда вызывать компилятор с опцией `-v`.

Вы можете запустить двоичный файл, полученный после компиляции, с помощью утилиты `trace` (чтобы увидеть каждый вызов функции и коды его результата). В утилите `trace` используются вызовы функции `debug`. Некоторые программисты предпочитают использовать в кодах вызовы функции `write`, чтобы проверить значение переменных или логику программы.

Если JavaScript выполняется с помощью LiveWire, то при этом создаются некоторые объекты, которыми можно управлять программно. Объект `request` содержит методы доступа к компонентам запроса HTTP, в том числе такие, которые в сценарии CGI передаются через переменные окружения, как-то `request.ip` и `request.agent`. В объекте `request` имеются переменные для полей каждой формы, а также URL.

Предопределенный объект `server` содержит другие члены, которые заменяют переменные окружения CGI, такие как `hostname`, `host` и `port`.

Для запоминания состояния пользователя между запросами в LiveWire используется объект `client`. С помощью файлов cookies Netscape или других статических механизмов сохранения информации можно написать приложение для того, чтобы запомнить выбор, сделанный пользователем в запросах. Метод `client.expiration(`*секунды*`)` указывает системе, что по истечении указанного времени (в секундах) бездействия необходимо удалить объект.

Виртуальная машина Java

В архитектуре всех новых серверов Netscape имеется виртуальная машина Java, которая позволяет переносить приложения из одной системы в другую. Ранее разработчикам приходилось писать сценарий CGI для UNIX, а затем приспосабливать его для NT. Технология Netscape позволяет создавать только одну версию программы — на JavaScript. Эта программа будет вызывать виртуальную машину Java независимо от того, какое оборудование и операционная система используется — UNIX, Windows NT или Windows 95.

Пример базы данных

В этом разделе мы рассмотрим простой пример приложения с использованием LiveWire Pro. Это приложение начинается с файла `start.htm`, показанного в листинге 36.1, в качестве начальной страницы и с файла `home.htm` из листинга 36.2 в качестве страницы, принятой по умолчанию.

Листинг 36.1. *start.htm* — Подключение к базе данных с использованием JavaScript (перевод)

```
<html>
<head>
    <title> Начало приложения оптовой торговли книгами </title>
</head>
<body>
<server>
if(!database connected())
    database.connect("INFORMIX", "myserver",
        "mmorgan", "ASecretWord", "booksDemo")
if(database.connected())
    write("Ошибка: Не могу подключиться к базе данных.")
else {
    redirect("home.htm")
}
</server>
</body>
</html>
```

Листинг 36.2. *home.htm* — Отправная страница, дающая пользователю доступ к возможностям приложения (перевод)

```
<html>
<head>
    <title> Приложение оптовой торговли книгами </title>
    <meta name="GENERATOR" content="Mozilla/2.01Gold (Win32)">
</head>
<body>
<hr>
<h1> Функции управления </h1>

<ul>
<li><a href="invent.htm">Показать список книг</a> </li>

<li><a href="addTitle.htm">Добавить название</a></li>

<li><a href="delTitle.htm">Удалить название</a></li>
```

```
<li><a href="sales.htm">Осуществить продажу</a></li>
</ul>

</body>
</html>
```

На рис. 36.9 показана начальная страница этого приложения.

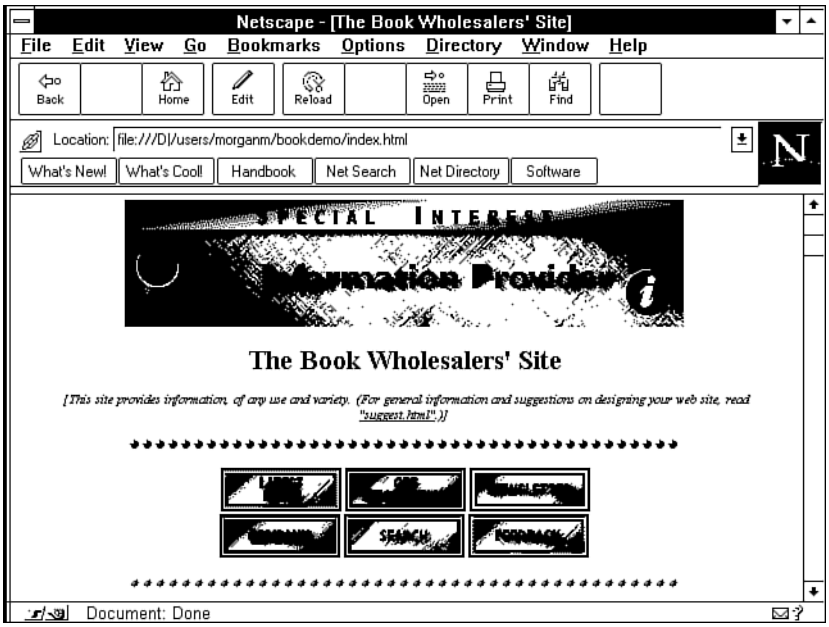

Рис. 36.9. Приложение позволяет добавлять и удалять названия книг, вести инвентарную ведомость и продавать книги

Пользователь имеет возможность создать в базе данных перечень наименований книг. В листинге 36.3 показано, как была создана эта возможность, а на рис. 36.10 вы видите полученный результат.

Листинг 36.3. *invent.htm* — Показ активной инвентарной ведомости (перевод)

```
<html>
<head>
    <title> Инвентарная ведомость </title>
    <meta name="GENERATOR" content="Mozilla/2.01Gold (Win32)">
</head>
<body>
<server>
    database.SQLTable("SELECT isbn,title, author,
        publishers. pubname, quantity On Hand FROM books,
        publishers WHERE books.publisherID = publish-
ers.publisherID");
</server>
<p>
<a href="home.htm">Home</a>
</p>
</body>
</html>
```

Пользователь выбирает страницу `Addtitle.htm` (листинг 36.4), и заполняет форму для добавления нового названия книги. Обратите внимание, что на этой странице создается список `<SELECT>` из записей базы данных, как показано на рис. 36.11.

Рис. 36.10. На странице `invent.htm` создается список всех книг в базе данных

Листинг 36.4. *Addtitle.htm* — Добавление новой книги в инвентарную ведомость (перевод)

```
<html>
<head>
    <title> Добавление новой книги </title>
    <meta name="GENERATOR" content="Mozilla/2.01Gold (Win32)">
</head>
<body>
<h1>Добавление новой книги</h1>
<p>Примечание: <b>Все</b>поля являются обязательными для добавления
новой книги.
<form method="post" action="add.htm"></p>
<br>Название:
<br>,input type="text" name="title" size="50">
<br>ISBN:
<br><input type="text" name="isbn" size="10">
<br>Розничная цена:
<br><INPUT TYPE="text" name="retailPrice" size="6">
<br>Издательство
<SELECT NAME="publisherID">
<SERVER>
publisherCursor = database.cursor("SELECT id, name FROM publishers
```

```
ORDER BY name");
while (publisherCursor.next())
{
    write("<OPTION
Value="+publisherCursor.id+">"+publisherCursor.name);
}
</SERVER>
</SELECT>
<BR>
<input type="submit" value="Enter">
<input type="reset" value="Clear">
</form>
<p><a href="home.htm">Home</a> </p>
</body>
</html>
```

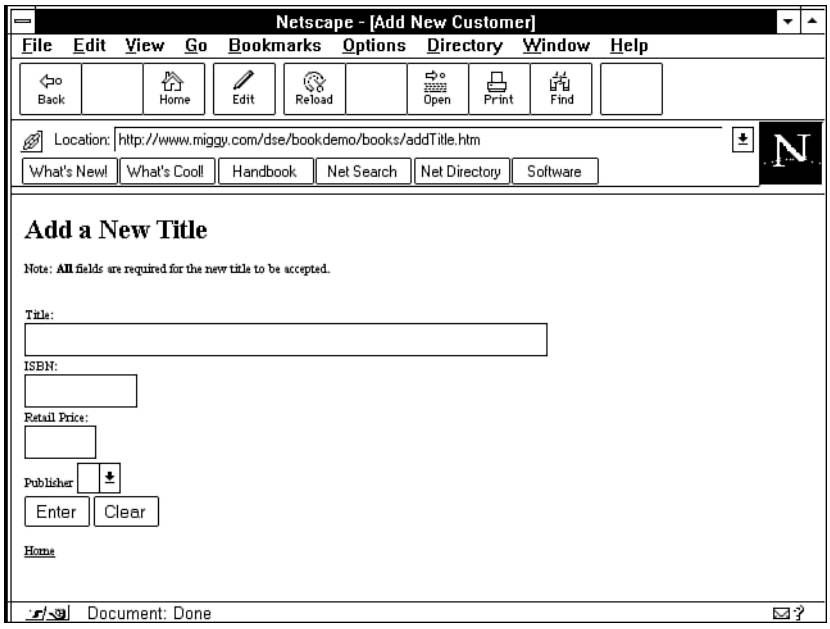

Рис. 36.11. На странице `Addtitle.htm` пользователь вводит данные о новой книге

Когда пользователь заполнит страницу `Addtitle.htm`, управление передается странице `Add.htm`, приведенной в листинге 36.5, которая выполняет вставку записи в базу данных. Затем управление опять возвращается странице `Addtitle.htm`.

Листинг 36.5. *Add.htm* — Завершение процесса добавления новой книги (перевод)

```
<html>
<head>
    <title> Добавление книги в базу данных </title>
    <meta name="GENERATOR" content="Mozilla/2.01Gold (Win32)">
</head>
<body>
<server>
    cursor = database.cursor("SELECT * FROM books",TRUE);
    cursor.isbn = request.isbn;
```

```
    cursor.title = request.title;
    cursor.retailPrice = request.retailPrice;
    cursor.publisherID = request.publisherID;
    cursor.quantity_on_hand = 0;
    cursor.updateRow(books);
    redirect("addTitle.htm")
</server>
</body>
</html>
```

Если вы перейдете на страницу `Deltitle.htm`, то увидите список названий всех имеющихся книг, созданный в процессе работы базы данных. Вы можете щелкнуть на коде ISBN, чтобы удалить книгу из базы данных. В листинге 36.6 показана эта страница, а на рис. 36.12 показано то, что видит пользователь на экране.

Листинг 36.6. *Deltitle.htm* — Пользователь готовится к удалению книги из списка (перевод)

```
<html>
<head>
    <title> Удаление книги </title>
</head>
<body>
<server>
cursor = database.cursor(SELECT isbn, title,
     retailPrice, publishers.name FROM books,
     publishers WHERE books.publisherID =
     publishers.ID ORDER BY isbn");
</server>
<table border>
<caption>
<center><p><b><font SIZE=+1>Книги отсортированы по
ISBN</font></b></p></center>
<center><p><b><font SIZE=+1>Щелкните на ISBN для удаления
книги</font></b></p></center>
</caption>
<tr>
<th>ISBN</th>
<th>Название</th>
<th>Розничная цена</th>
<th>Издательство</th>
</tr>
<caption>
<center><p>
<server>
while(cursor.next())
{
    write("<TR><TD><A HREF='remove.htm?isbn='"
        +cursor.isbn+"</A></TD><TD>"+cursor.title+
        "</TD><TD>"+cursor.retailPrice+"</TD><TD>"+
        cursor.name+"</TD></TR>");
}
</table>
</body>
</html>
```

Страница `remove.htm` обновляет базу данных. В листинге 36.7 показан код этой страницы

Листинг 36.7. *Remove.htm* — Удаление книги из базы данных

```
<html>
<head>
<title> Удаление записи из базы данных </title>
</head>
<server>
if(request.isbn !=null)
{
    cursor = database.cursor ("SELECT * FROM books
        WHERE isbn ="+ request.isbn,TRUE);
    cursor.deleteRow(books)
}
redirect("delTitle,htm");
</server>
</body>
</html>
```

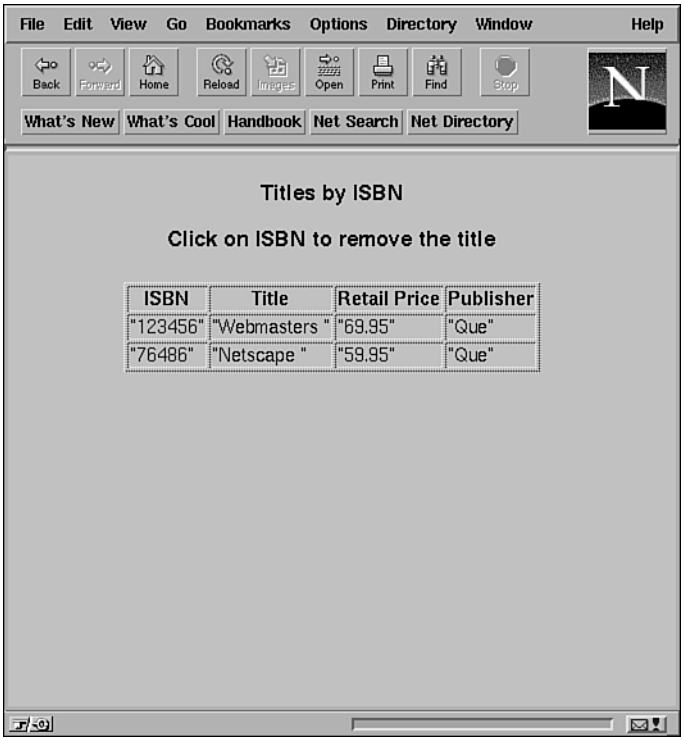

Рис. 36.12. Пользователь вбирает книгу, которую необходимо удалить. Этот список создается на сервере из базы данных с помощью JavaScript

Чтобы добавить в инвентарную ведомость информацию о проданных книгах, сотрудник склада переходит к странице `Sales.htm`. В листинге 36.8 показан код этой страницы, а саму страницу вы видите на рис. 36.13.

Листинг 36.8. *Sales.htm* — Продажа книг (перевод)

```
<html>
<head>
    <title> Продажа книг </title>
</head>
<body>
<h1>Продажа книг<h1>
<p>Примечание: <b>Все</b> поля являются
    обязательными для оформления продажи.
<form method="post" action="sell.htm"></p>
<br>ISBN:
<br><input type="text" name="isbn" size="10">
<br>Количество экземпляров:
<br><INPUT TYPE="text" name="copies" size="6">
<BR>
<input type="submit" value="Enter">
<input type="reset" value="Clear">
</form>
<p><a href="home.htm">Home</a> </p>
</body>
</html>
```

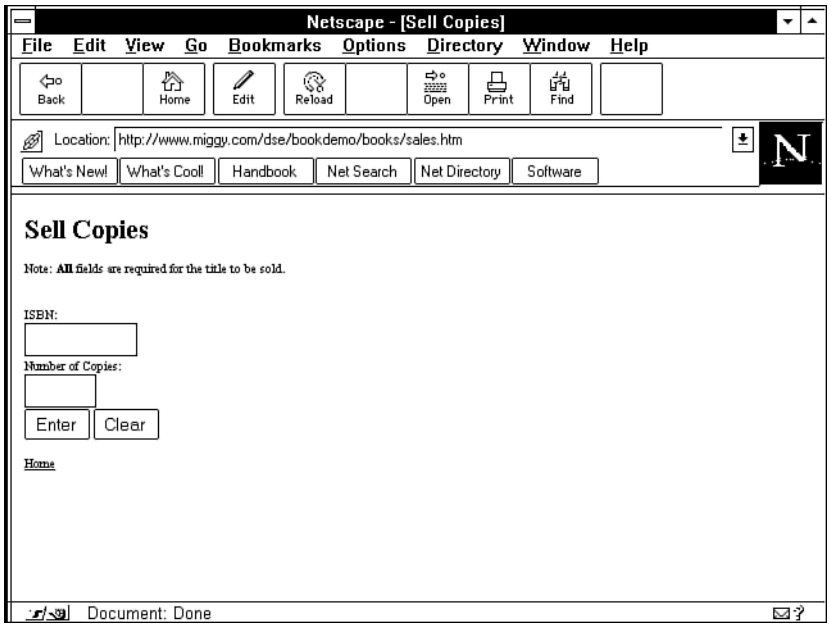

Рис. 36.13. Страница для ввода информации о проданных книгах

В листинге 36.9 показано, как подтвердить транзакцию.

Листинг 36.9. *sell.htm* — Подтверждение транзакции <HTML> (перевод)

```
<HEAD>
<TITLE>Продажа книг</TITLE>
</HEAD>
<BODY>
cursor = database.cursor("SELECT title, isbn, retailPrice,
        publishers.name, quantityOnHand FROM books, publishers
        WHERE isbn=" + request.isbn +" AND
        publishers.ID = books.publisherID");
if (cursor.next())
{
    if (cursor.quantityOnHand > request.quantity)
    {
        write ("<FORM ACTION=sold.htm METHOD=GET>");
        write ("<P>Confirm sale of <STRONG>" + request.copies +
            </STRONG> of <BR>" + cursor.title + "<BR>ISBN " +
            cursor.isbn + "<BR>Retail Price " +
            cursor.retailPrice + "<BR>Publisher " +
            cursor.name</P>");
        write ("<INPUT TYPE=submit NAME=submit VALUE=Yes>");
        write ("<INPUT TYPE=button NAME=home VALUE=No
            onClick='redirect("home.htm");'>");
        write ("<INPUT TYPE=hidden NAME=isbn VALUE=" +
            request.isbn + ">");
        write ("<INPUT TYPE=hidden NAME=quantity VALUE=" +
            request.quantity + ">");
        write ("</FORM>");
    }
    else
        write ("<P>На складе имеется только " + cursor.quantityOnHand
+
            " экземпляров.</P>");
}
else
{
    write ("<P>ISBN " + request.isbn + " нет в базе данных.</P>);
</BODY>
</HTML>
```

Страница `sold.htm` выполняет обновление базы данных. В листинге 36.10 показан соответствующий код, а на рис. 36.14 — внешний вид страницы.

Листинг 36.10. *sold.htm* — Обновление базы данных с учетом информации о проданных книгах (перевод)

```
<HTML>
<HEAD>
<TITLE>Продажа книг</TITLE>
</HEAD>
<BODY>
<SERVER>
cursor = database.cursor("SELECT * FROM BOOKS
        WHERE isbn=" + request.isbn,TRUE);

//move onto selected row
```

```
cursor.next();
cursor.quantityOnHand = cursor.quantityOnHand
                        - request.quantity;
cursor.updateRow(books);
</SERVER>
<P>
<H1>Операция завершена</H1>
<P>
<server>
write ("Продано " + request.quantity + " книг " + request.isbn +
".");
<server>
</P>
<A HREF="home.htm">Home</A>
</BODY>
</HTML>
```

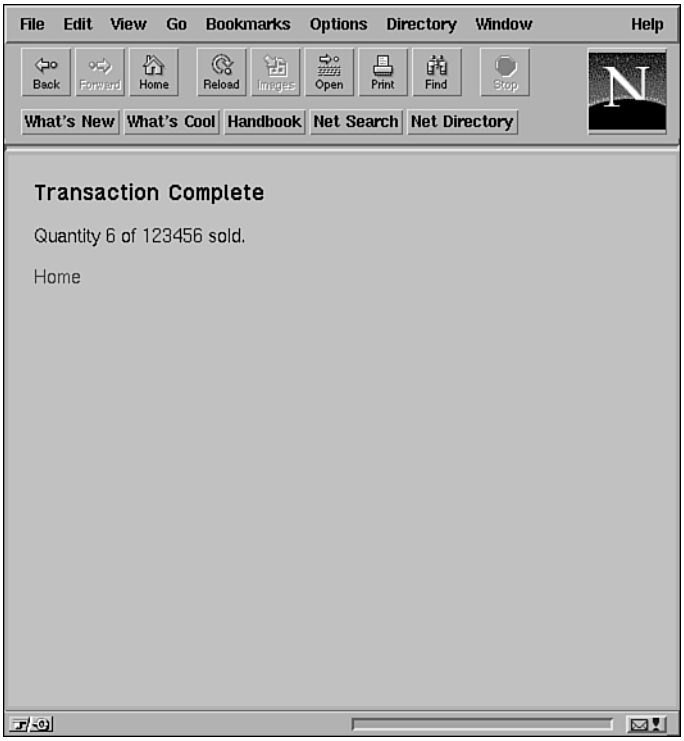

Рис. 36.14. Подтверждение изменения базы данных