Содержание

Глава 17

Доступ к данным с помощью ODBC и JDBC

В ЭТОЙ ГЛАВЕ...

- Что такое ODBC
- Компоненты ODBC
- Использование ODBC в среде "клиент/сервер"
- Использование ODBC в Интернете
- Использование ODBC в локальной сети
- Использование JDBC

С каждым годом компьютеры все активнее и активнее подключаются к сетям, как локальным, так и глобальным. Как следствие, возникла необходимость в обеспечении совместного доступа к базам данных по сети, однако этому препятствует несовместимость системного программного обеспечения и приложений, работающих на разных компьютерах. Важными этапами на пути преодоления такой несовместимости стал стандарт SQL.

К сожалению, стандарт SQL не реализован на практике в чистом виде. Производители СУБД, утверждающие, что их продукты совместимы с международным стандартом SQL, зачастую включают в свои реализации расширения, несовместимые с продуктами *других* производителей. Производители не склонны отказываться от своих расширений, так как покупатели привыкли

к ним и зависят от них. Крупным организациям для совместного использования различных реализаций СУБД необходим другой подход, не требующий от производителей приводить их продукты к "наименьшему общему знаменателю". Этот подход и реализует открытый интерфейс доступа к базам данных — ODBC (Open DataBase Connectivity).

ODBC

ODBC — это стандартный интерфейс между базой данных и приложением, взаимодействующим с ней. Наличие подобного стандарта позволяет любому приложению на клиентском компьютере получать доступ к любой базе данных на сервере с помощью SQL. Единственное требование заключается в том, чтобы клиентская и серверная части поддерживали стандарт ODBC. ODBC 4.0 — текущая версия стандарта.

Приложение получает доступ к конкретной базе данных, используя специально разработанный для нее *драйвер* (в данном случае драйвер ODBC). Клиентская часть драйвера, работающая напрямую с приложением, должна строго соответствовать стандарту ODBC. Приложению все равно, какая СУБД установлена на сервере. Серверная часть драйвера адаптирована к конкретной базе данных. Благодаря такой архитектуре не только не нужно настраивать приложения на определенную СУБД — им даже не обязательно знать, какая именно СУБД используется. Драйвер скрывает различия между серверами баз данных.

Интерфейс ODBC

Интерфейс ODBC — это унифицированный набор определений, необходимых для организации взаимодействия приложения и базы данных. Каждое такое определение принимается в качестве стандарта. Интерфейс ODBC определяет следующее:

- библиотеку вызовов функций;
- стандартный синтаксис SQL;
- стандартные типы данных SQL;
- стандартный протокол соединения с базой данных;
- стандартные коды ошибок.

Вызовы функций ODBC обеспечивают подключение приложения к серверу базы данных, выполнение инструкций SQL и возврат результатов приложению.

СОВЕТ

Чтобы выполнить какое-либо действие с базой данных, в качестве аргумента функции ODBC необходимо указать соответствующую инструкцию SQL. При условии использования стандартного для ODBC синтаксиса SQL результат выполнения этой функции не будет зависеть от того, какая база данных установлена на сервере.

Компоненты ODBC

Интерфейс ODBC состоит из четырех функциональных компонентов, именуемых *уровнями ODBC*. Благодаря каждому из них достигается гибкость ODBC, позволяющая взаимодействовать любым ODBC-совместимым клиентам и серверам. Между пользователем и данными, которые он хочет получить, находятся четыре уровня интерфейса ODBC.

- **Приложение.** Это та часть интерфейса ODBC, с которой непосредственно работает пользователь. Естественно, приложения могут быть не только в ODBC-совместимых системах. Однако включение приложения в интерфейс ODBC имеет определенный смысл. Приложение должно взаимодействовать с источником данных при посредничестве ODBC и подключаться с помощью диспетчера драйверов ODBC в строгом соответствии со стандартом ODBC.
- **Диспетчер драйверов.** Это динамически подключаемая библиотека (DLL), обычно поставляемая компанией Microsoft. Она загружает необходимые драйверы для системных источников данных (возможно, нескольких) и направляет вызовы функций приложений с помощью драйверов к соответствующим источникам данных. Диспетчер драйверов непосредственно обрабатывает некоторые вызовы функций ODBC, а также перехватывает и обрабатывает определенные типы ошибок. Несмотря на то что стандарт ODBC внедрила компания Microsoft, он стал общепринятым (его приняли даже бескомпромиссные сторонники систем с открытым исходным кодом).
- **Драйвер DLL.** В связи с тем что источники данных могут различаться, причем в некоторых случаях весьма существенно, необходим способ преобразования стандартных вызовов функций ODBC в языковый код конкретного источника данных. Этим и занимается драйвер DLL. Каждый драйвер получает вызовы функций через стандартный интерфейс ODBC и переводит их в код, понятный источнику данных. Как только источник данных готов вернуть результат, драйвер в обратном порядке преобразует его в стандартный для ODBC вид. Драйвер является основным элементом, который позволяет ODBC-совместимому приложению управлять структурой и содержимым ODBC-совместимого источника данных.

» **Источник данных.** Существует множество различных источников данных. Таким источником может быть база данных на основе реляционной СУБД, находящаяся на одном компьютере с приложением, или база данных на удаленном компьютере. В качестве источника данных может выступать файл, хранящийся вне СУБД, доступ к данным которого реализован *индексно-последовательным методом* (indexed sequential access method — ISAM). Такие файлы могут быть расположены как на локальном, так и на удаленном компьютере. Для каждого вида источников данных требуется свой драйвер.

ODBC в среде "клиент/сервер"

В среде "клиент/сервер" интерфейс между клиентской и серверной частями называется API (Application Programming Interface — интерфейс прикладного программирования). API может быть как специальным, так и стандартным. *Специальным* называется интерфейс, разработанный для взаимодействия с конкретной серверной частью. Программным кодом, который формирует этот интерфейс, является драйвер, и в специальной системе он называется собственным драйвером. *Собственный драйвер* оптимизирован для работы с определенной клиентской частью и связанной с ней серверной частью источника данных. В связи с тем что собственные драйверы настроены для работы как с приложением, так и с СУБД, они передают команды и информацию достаточно быстро и с минимальными задержками.


СОВЕТ

Если система "клиент/сервер" рассчитана на доступ к конкретному источнику данных и заведомо не будет использовать другой, лучше воспользоваться собственным драйвером из поставки СУБД. С другой стороны, если системе требуется обеспечивать доступ к данным, хранящимся в различных форматах, использование ODBC API избавит разработчика от выполнения огромного объема ненужной работы.

Драйверы ODBC оптимизированы для работы с определенными источниками данных серверной части, однако все они имеют одинаковый внешний интерфейс с диспетчером драйверов. Любой драйвер, не оптимизированный для работы с конкретным клиентом, скорее всего, проиграет в быстродействии собственному драйверу, который специально разработан для данного клиента. Основным недостатком первого поколения драйверов ODBC была их низкая производительность по сравнению с собственными драйверами. Однако последние измерения показали, что производительность драйверов ODBC 4.0 вполне сравнима с производительностью собственных драйверов.

На сегодняшний день технология достигла того уровня, когда уже не нужно жертвовать производительностью ради преимуществ стандартизации.

ODBC в Интернете

Операции с базами данных в Интернете кое в чем серьезно отличаются от операций с базами данных в среде "клиент/сервер". Самое заметное отличие, с точки зрения пользователя, заключено в клиентской части системы, содержащей интерфейс пользователя. В среде "клиент/сервер" интерфейс пользователя — это часть приложения, которая связывается с источником данных на сервере посредством ODBC-совместимых инструкций SQL. В веб-среде клиентская часть системы по-прежнему находится на локальном компьютере, но взаимодействует с источником данных на сервере с помощью стандартного протокола HTTP.

Любой пользователь, располагающий соответствующим клиентским программным обеспечением (и соответствующими полномочиями), может получить доступ к данным, находящимся в Интернете. Это означает, что можно создать приложение на своем рабочем компьютере, а позже получить доступ к нему со своего мобильного устройства. На рис. 17.1 показаны основные различия между системой "клиент/сервер" и системой, созданной с помощью веб-технологий.

Серверные расширения

В системе, созданной на основе веб-технологий, клиентский компьютер и веб-сервер взаимодействуют с помощью протокола HTTP. *Серверное расширение* — это компонент системы, который преобразует команды, передаваемые по сети, в ODBC-совместимый SQL-код, после чего сервер базы данных связывается с источником данных и выполняет этот код. В обратном направлении источник данных пересылает результат запроса серверу базы данных и далее — серверному расширению, которое преобразует результат запроса к виду, понятному веб-серверу. Затем данные отсылаются клиентскому компьютеру по Интернету и отображаются у пользователя. На рис. 17.2 приведена схема такого взаимодействия.

Клиентские расширения

Такие приложения, как Mircosoft Access 2016, предназначены для обработки данных, которые хранятся либо локально на компьютере пользователя, либо на сервере, расположенном в локальной или глобальной сети, либо в облачных хранилищах Интернета. Хранилище, предоставляемое компанией Microsoft,

называется OneDrive. Доступ к приложению в облаке можно также получить с помощью браузера. Браузеры создаются и оптимизируются для организации простого и интуитивно понятного интерфейса доступа к веб-серверам любых типов. Наиболее популярные браузеры, такие как Google Chrome, Mozilla Firebox, Microsoft Internet Explorer и Apple Safari, изначально не предназначались (и не оптимизировались) для использования в качестве клиентской части базы данных. Чтобы добиться требуемого уровня взаимодействия с базой данных в Интернете, необходимы дополнительные функциональные возможности. Для обеспечения этих возможностей были разработаны различные *клиентские расширения*, которые включают элементы управления ActiveX, аплеты Java и сценарии. Расширения взаимодействуют с сервером с помощью протокола HTTP, используя стандартный язык — HTML. Любой HTML-код, получающий доступ к базе данных, перед отправкой источнику данных преобразуется серверным расширением в ODBC-совместимый SQL-код.

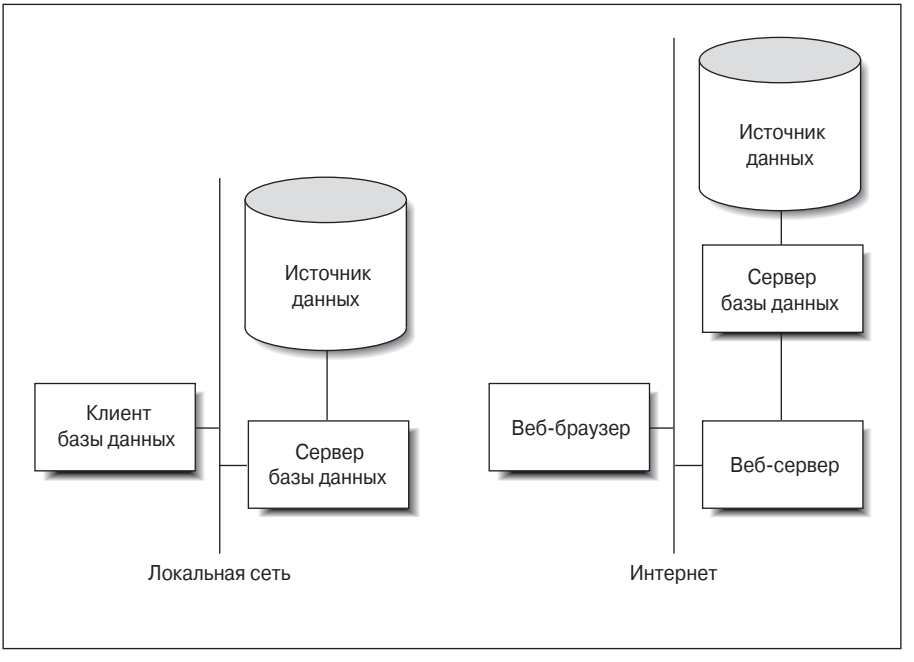

Рис. 17.1. Система "клиент/сервер" в сравнении с веб-ориентированной системой

Элементы управления ActiveX

Элементы управления Microsoft ActiveX работают с Microsoft Internet Explorer — одним из самых популярных браузеров в мире (хотя его позиции значительно пошатнулись с ростом популярности таких браузеров, как Google Chrome и Mozilla Firefox).

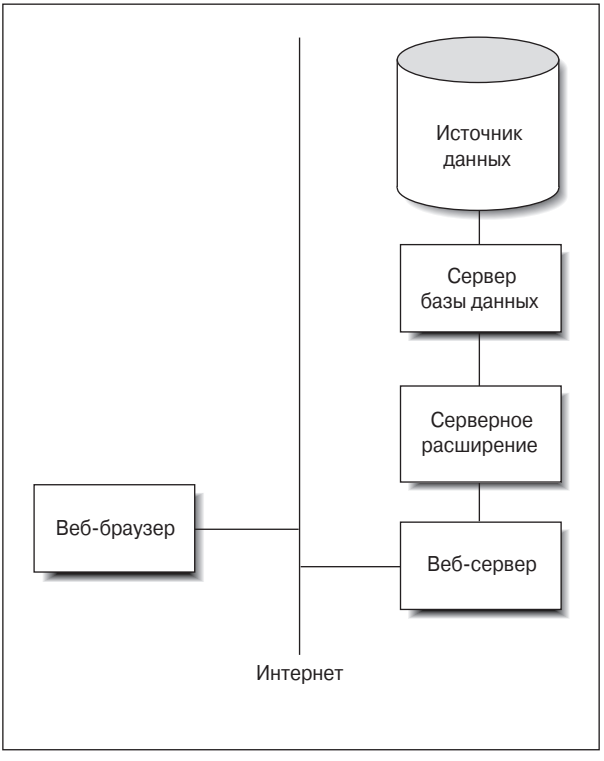

Рис. 17.2. Веб-ориентированная СУБД с серверным расширением

Сценарии

Сценарии — наиболее гибкий инструмент создания клиентских расширений. Использование языка сценариев, такого как JavaScript или Microsoft VBScript, позволяет максимально контролировать происходящее на клиентском компьютере. С помощью сценариев можно проверять достоверность данных, вводимых в поля формы, и отбраковывать неправильно заполненные формы еще на клиентском компьютере. Это сэкономит ваше время и нагрузку на интернет-канал. Безусловно, контроль данных можно также выполнять на сервере путем применения ограничений к значениям элементов данных. Как и аплеты Java, сценарии встраиваются в веб-страницу и выполняются при ее открытии пользователем.

ODBC в локальной сети

Интранет — это локальная или региональная сеть, работающая как упрощенная версия Интернета. Поскольку вся сеть принадлежит одной

организации, как правило, отсутствует необходимость в применении комплексных мер безопасности, таких как брандмауэры. Все инструменты, разработанные для создания веб-приложений, подходят также для создания приложений локальных сетей. ODBC работает в локальных сетях точно так же, как и в Интернете. При наличии нескольких различных источников данных клиенты, использующие браузеры и соответствующие клиентские и серверные расширения, могут взаимодействовать с этими источниками посредством SQL-кода, передаваемого с помощью HTTP и ODBC. Драйвер ODBC преобразует SQL-код в собственный язык команд базы данных и выполняет его.

JDBC

Интерфейс Java-приложений для взаимодействия с базами данных (Java DataBase Connectivity — JDBC) имеет много общего с ODBC, но в то же время содержит несколько существенных отличий. Одно из отличий явствует из названия. Как и ODBC, JDBC — универсальный интерфейс доступа к базам данных, не зависящий от источника данных на сервере. Различие состоит в том, что клиентское приложение для JDBC может быть написано только на Java, а не на каком-то другом языке программирования (например, C++ или Visual Basic). Еще одно отличие состоит в том, что Java и JDBC от начала и до конца разрабатывались для использования в Интернете.

Java — это язык программирования (похожий на C++), разработанный компанией Sun Microsystems специально для создания клиентских веб-ориентированных программ. После установления соединения между клиентской машиной и сервером соответствующий аплет Java загружается на компьютер клиента, где и выполняется. Аплет, внедренный на веб-страницу, предоставляет функции взаимодействия с базой данных, реализуя гибкий доступ клиента к данным на сервере. На рис. 17.3 показан механизм работы веб-приложения с базой данных с использованием аплета Java, запущенного на клиентской машине.

Аплет — это небольшое приложение, находящееся на сервере. Когда клиент подключается по Интернету к серверу, аплет загружается на клиентский компьютер и запускается. Аплеты Java разработаны таким образом, чтобы запускаться в своей *песочнице*. Песочница — это определенное (изолированное) место в памяти клиентского компьютера, где выполняются аплеты Java. Аплет не может взаимодействовать с чем-нибудь вне песочницы. Такая архитектура позволяет защитить клиентский компьютер от потенциально опасных аплетов, которые могут получить доступ к конфиденциальной информации или нанести серьезный вред данным.

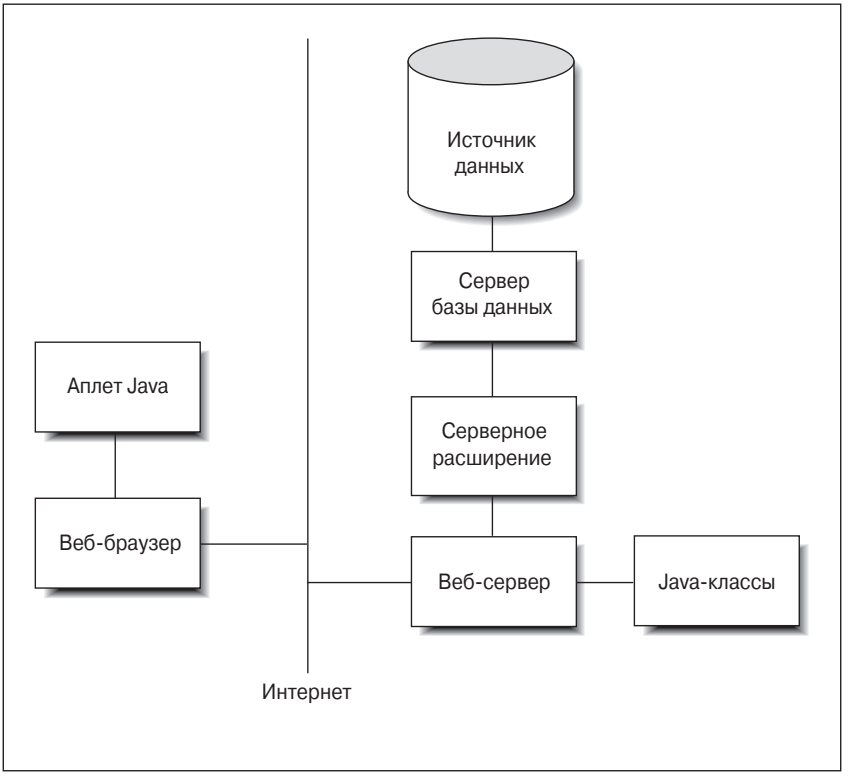

Рис. 17.3. Веб-приложение, предназначенное для работы с базой данных с использованием аплета Java

Главное достоинство аплетов Java заключается в том, что они постоянно обновляются. Поскольку аплеты загружаются с сервера при каждом использовании (а не остаются постоянно на машине клиента), клиент всегда снабжен последней версией аплета, доступной на момент его запуска.


СОВЕТ

Если вы отвечаете за поддержку сервера своей организации, то вам никогда не придется волноваться о совместимости с любым из клиентов при обновлении программного обеспечения сервера. Просто убедитесь в том, что загружаемый аплет Java совместим с новой конфигурацией сервера, и если все веб-браузеры поддерживают аплеты Java, то они автоматически также станут совместимыми с сервером. Java — это полнофункциональный язык программирования, позволяющий создавать надежные приложения доступа к базам данных в любых конфигурациях “клиент/сервер”. При таком использовании доступ Java-приложений к базам данных посредством JDBC подобен доступу к данным приложений C++ посредством ODBC. В то же

время при использовании в Интернете (или в локальной сети) работа приложений Java в корне отличается от работы приложений C++.

Когда система функционирует в Интернете, условия ее работы отличаются от условий в среде "клиент/сервер". Клиентская часть приложения, которая работает с Интернетом, — это браузер с минимальными вычислительными возможностями. Эти возможности должны быть расширены, чтобы переложить на клиента часть работы с базой данных, и такое расширение функций обеспечивают аплеты Java.


ВНИМАНИЕ!

Загрузка данных с неизвестного сервера связана с рядом потенциальных опасностей. Если же вы загружаете Java-аплет, то уровень опасности резко снижается, хотя и не до нуля. Стоит с осторожностью относиться к сомнительному серверу и не позволять исполняемому коду беспрепятственно "заходить" на ваш компьютер.

Как и ODBC, JDBC передает SQL-инструкции от клиентской части приложения (аплета), запускаемого на компьютере клиента, к источнику данных на сервере. Также JDBC служит для передачи результатов выполнения запросов или сообщений об ошибках от источника данных назад приложению. Выгода от использования JDBC заключается в том, что разработчик аплетов может использовать стандартный интерфейс JDBC, не заботясь о том, какая база данных находится на сервере. JDBC выполняет все преобразования, необходимые для корректного двухстороннего взаимодействия. Несмотря на то что интерфейс JDBC предназначен для работы в веб-среде, он может также работать в среде "клиент/сервер", обеспечивая посредничество для взаимодействия приложения, написанного на языке Java, с серверной частью базы данных.