Driver types. Making connection.

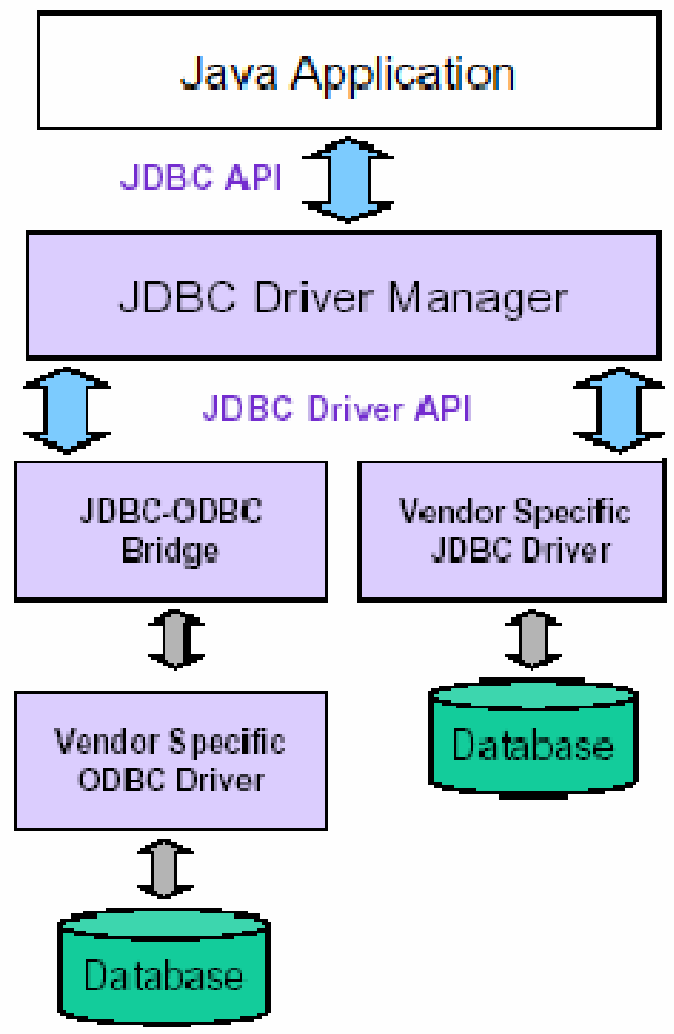
JDBC 4.0

JDBC

JDBC (Java DataBase Connectivity) – стандартный прикладной интерфейс (API) языка Java для организации взаимодействия между приложением и СУБД.

Это взаимодействие осуществляется с помощью драйверов JDBC, обеспечивающих реализацию общих интерфейсов для конкретных СУБД и конкретных протоколов.

Архитектура JDBC



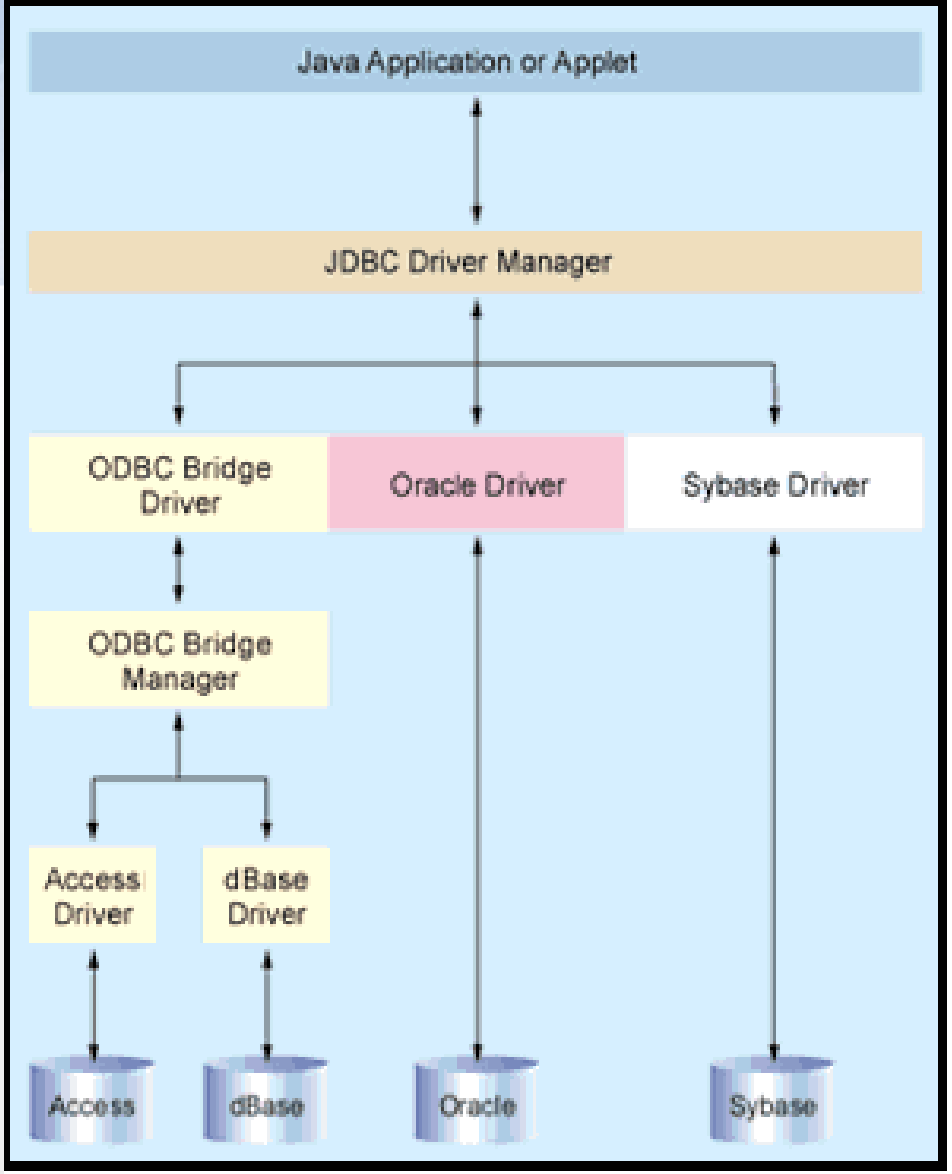


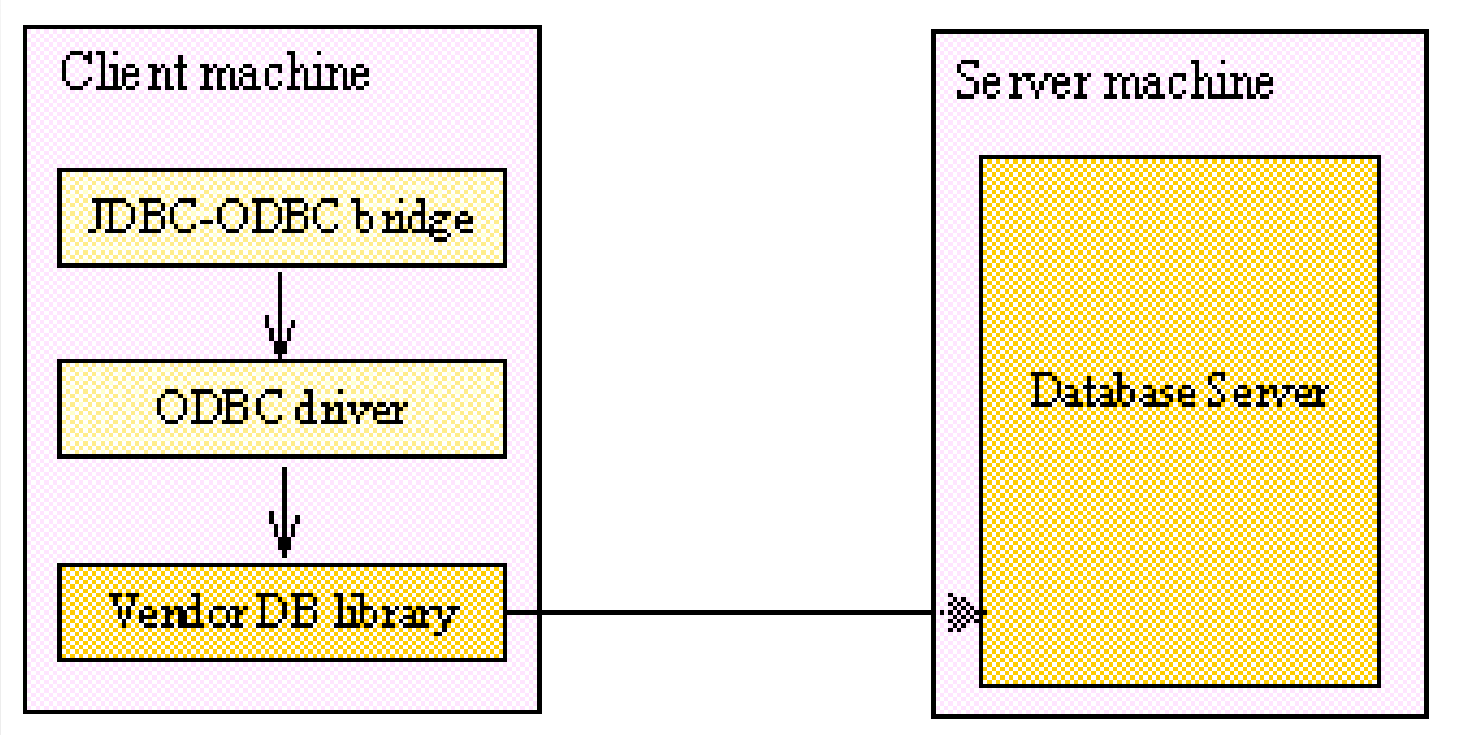
Figure 1. Anatomy of Data Access. The Driver Manager provides a consistent layer between your Java app and back-end database. JDBC works natively (such as with the Oracle driver in this example) or with any ODBC datasource.

Четыре типа драйверов

1. Драйвер, использующий другой прикладной интерфейс взаимодействия с СУБД, в частности ODBC (так называемый JDBC-ODBC – мост). Стандартный драйвер первого типа **sun.jdbc.odbc.JdbcOdbcDriver** входит в JDK.
2. Драйвер, работающий через внешние (native) библиотеки (т.е. клиента СУБД).
3. Драйвер, работающий по сетевому и независимому от СУБД протоколу с промежуточным Java-сервером, который, в свою очередь, подключается к нужной СУБД.
4. Сетевой драйвер, работающий напрямую с нужной СУБД и не требующий установки native-библиотек.

JDBC-ODBC бридж

Драйверы 1-го типа транслируют все вызовы JDBC в вызовы ODBC (Open Database Connectivity), с пересылкой всех данных в ODBC драйвер.



JDBC-ODBC бридж

Плюсы

- JDBC-ODBC бридж позволяет осуществить доступ практически с любой базе данных, поскольку драйверы ODBC существуют практически к любым СУБД.

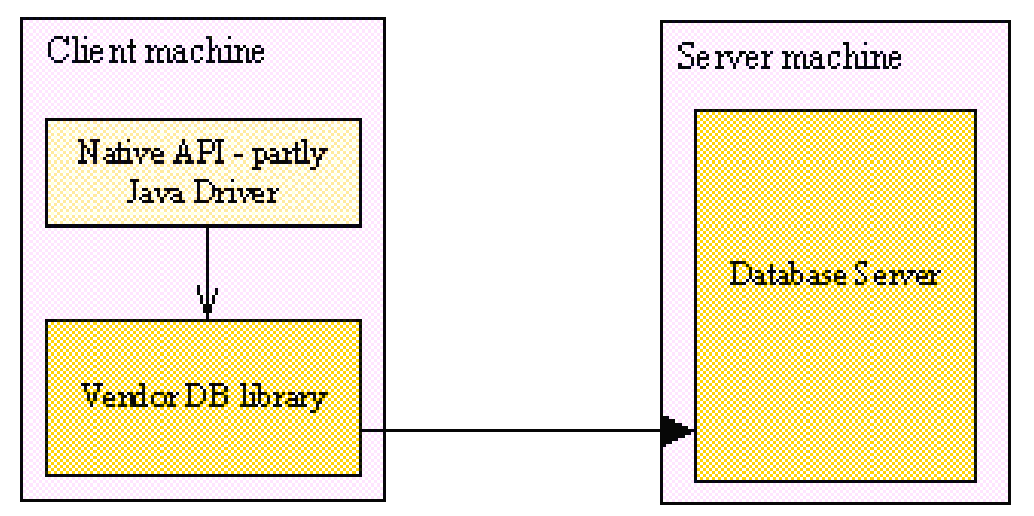
Минусы

- Низкая производительность. Драйверы 1-го типа могут не подходить для приложений большого масштаба.
- ODBC драйвер и специальное клиентское ПО должны быть установлены на клиентской машине, то есть отсутствует возможность использования Java апплетов в интранет среде.

Нативный-API/частичный Java драйвер

JDBC драйвер 2-го типа - нативный-API/частичный Java драйвер – переводит вызовы JDBC в вызовы специфичные к СУБД таких как например SQL Server, Informix, Oracle или Sybase.

Драйвер 2-го типа общается напрямую с сервером базы данных



Нативный-API/частичный Java драйвер

Плюсы

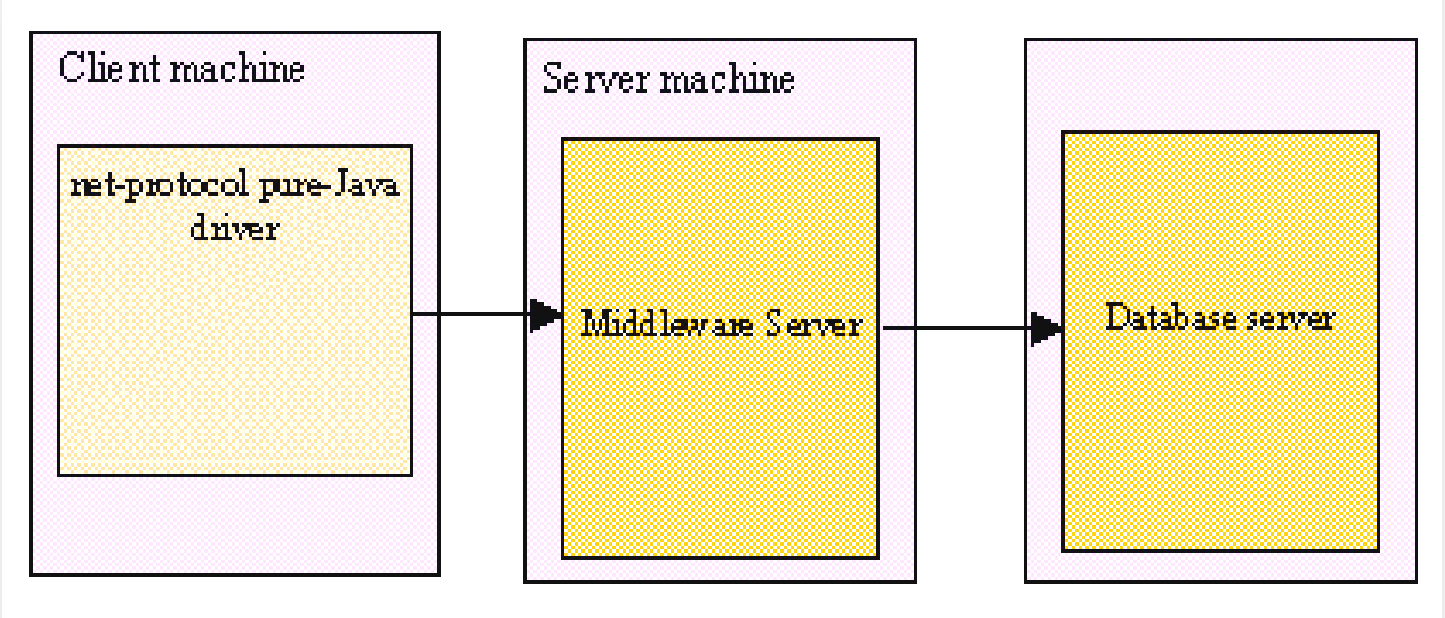
- Драйвер 2 типа как правило обеспечивает значительно большую производительность по сравнению с JDBC-ODBC бриджем.

Минусы

- Требуется установка на стороне клиента библиотек производителя СУБД. Следовательно, драйверы 2 типа не могут использоваться для Интернет приложений. Драйверы этого типа менее производительны, чем драйверы 3 и 4 типов.

Сетевой протокол/«чистый» Java драйвер

JDBC драйвер 3 типа – сетевой протокол/«чистый» Java драйвер – использует трехуровневую архитектуру. Если сервер среднего слоя написан на Java то он может использовать для трансляции JDBC драйверы 1 и 2 типов.



Сетевой протокол/«чистый» Java драйвер

Плюсы

- Данный драйвер является серверным, поэтому нет необходимости в установке библиотек вендора СУБД на клиентских машинах. Кроме того существует много способов для оптимизации производительности и масштабируемости.

Минусы

- Драйверы 3-го типа требуют кодирования на среднем слое функций специфичных к СУБД, кроме того, перебор записей в ResultSet может быть достаточно длительным, так как данные проходят через сервер приложений.

Нативный протокол/«чистый» Java драйвер

Нативный протокол/«чистый» Java драйвер (JDBC драйвер 4-го типа) конвертирует вызовы JDBC в специфический протокол вендора СУБД, так что клиентские приложения могут напрямую обращаться с сервером базы данных.

Драйверы 4-го типа полностью реализуются на Java с целью достижения платформенной независимости и устранения проблем администрирования и развертывания.

Нативный протокол/«чистый» Java драйвер

Плюсы

- Поскольку драйверам 4-го типа не требуется транслировать вызовы к базе данных в ODBC или нативный интерфейс вызова или же перенаправлять на другой сервер, то производительность этих драйверов обычно довольно хорошая.

Минусы

- При использовании драйверов 4-го типа, пользователю необходимо использовать различные драйвера для каждой базы данных.

Загрузка драйвера

Загрузка класса драйвера базы данных при отсутствии экземпляра этого класса.

Class.forName(driverName);

- **String driverName = "com.mysql.jdbc.Driver ";**

для СУБД MySQL,

- **String driverName = "sun.jdbc.odbc.JdbcOdbcDriver";**

для СУБД MSAccess или

- **String driverName = "org.postgreesql.Driver";**

для СУБД PostgreeSQL.

Установка соединения с БД.

- Для установки соединения с БД вызывается статический метод getConnection() класса DriverManager.
- В качестве параметров методу передаются URL базы данных, логин пользователя БД и пароль доступа. Метод возвращает объект Connection.
- URL базы данных, состоящий из типа и адреса физического расположения БД, может создаваться в виде отдельной строки или извлекаться из файла ресурсов.

```
Connection cn = DriverManager.getConnection(
    "jdbc:mysql://localhost/my_db", "root", "pass");
```

Класс DriverManager предоставляет средства для управления набором драйверов баз данных.

- метод registerDriver() – регистрирует драйверы,
- метод getDrivers() возвращает список всех драйверов.

URL драйвера

URL служит для идентификации источника данных драйвером JDBC

jdbc:<subprotocol>:<subname>

Для MySQL

jdbc:mysql://<host>:<port>/<database>