Volume 7, Issue 5, May 2017 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Study on Implementation and Application of Java Database Connectivity Drivers

Ashima Gambhir
Amity School of Engineering & Technology, Amity University, Gurgaon,
Haryana, India

DOI: 10.23956/ijarcsse/SV7I5/0233

Abstract: ***This research paper aims to help in carrying out the study on Implementation and application of Java Database Connectivity drivers. With the maturity of network technology, the deterministic factor of a 'successful' web site does not as much depend on its connectivity to the Internet, rather the content of web pages plays a more important role. The inclusion of dynamic data, for example, has become an increasingly desirable feature of a web page. Web servers that support dynamic web pages are capable of retrieving data at the time of user requests. Rather than being statically included in the web page, these dynamically retrieved data may physically reside in various file systems or database servers across the Internet. In this Paper, I have reviewed the java database connectivity drivers and comparison among them. Four types of JDBC Drivers namely JDBC-ODBC, Native-API, Network Protocol and Thin Driver are being deeply reviewed. The benefit which is being incurred by these types has been analyzed and finally I have compared the Four types of JDBC drivers.***

Keywords: JDBC, ODBC, API, Java, MYSQL.

I. INTRODUCTION

Java is a high-level object-oriented programming language developed by the Sun Microsystems. It was only developed keeping in mind the consumer electronics and communication equipment's. It came into existence as a part of web application, web services and a platform independent programming language in the 1990s. Earlier, C++ was widely used to write object oriented programming languages, however, it was not a platform independent and needed to be recompiled for each different CPUs. A team of Sun Microsystems including Patrick Naughton, Mike Sheridan in the guidance of James Goslings decided to develop an advanced programming language for the betterment of consumer electronic devices. They wanted to make it new software based on the power of networks that can run on different application areas, such as computers and electronic devices.

Java Database Connectivity(JDBC) is an Application Programming Interface(API) used to connect Java application with Database. JDBC is used to interact with various type of Database such as Oracle, MS Access, My SQL and SQL Server. JDBC can also be defined as the platform-independent interface between a relational database and Java programming. It allows java program to execute SQL statement and retrieve result from database.

In this paper, we discuss our experience of developing JAVA applets that use various types of JDBC (JAVA Database Connectivity) drivers to connect to and then manipulate data stored in SQL (Structured Query Language) database servers. We first start with the discussion of general client/server model involving JDBC, and then introducing the procedure of setting up programming projects employing this approach in an undergraduate database course. Figure 1 explains the Java Database Connectivity architecture.

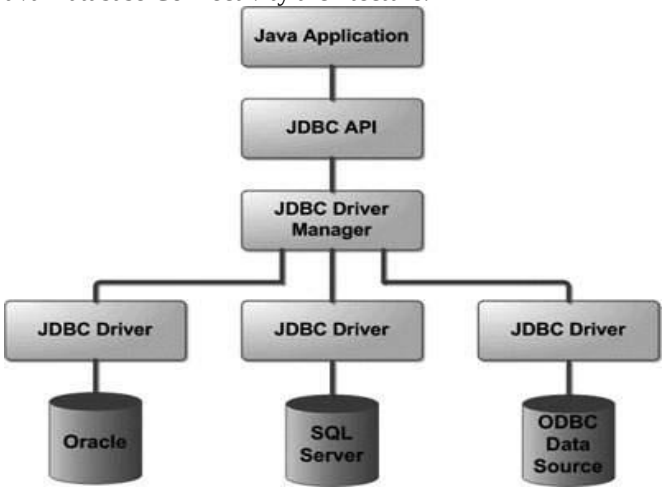

Fig1: JDBC Architecture

JDBC Features:

1) Connection Management.
2) Support for Large Objects.
3) Annotation in SQL Query.
4) Auto Loading of Driver Interface.
5) Better Exception Handling.

II. JDBC DRIVERS

JDBC Driver is required to process SQL requests and generate result. The following are the different types of driver available in JDBC.

2.1 Type 1 Driver or JDBC or ODBC Bridge

Type-1 Driver act as a bridge between JDBC and other database connectivity mechanism(ODBC). This driver converts JDBC calls into ODBC calls and redirects the request to the ODBC driver. The type 1 drivers rely on ODBC which reduces their portability and desirability. Moreover, querying through the ODBC Bridge slows down transactions. This type allow easy connectivity to all database supported by the ODBC Driver. Figure 2 explains the architecture of JDBC-ODBC bridge.

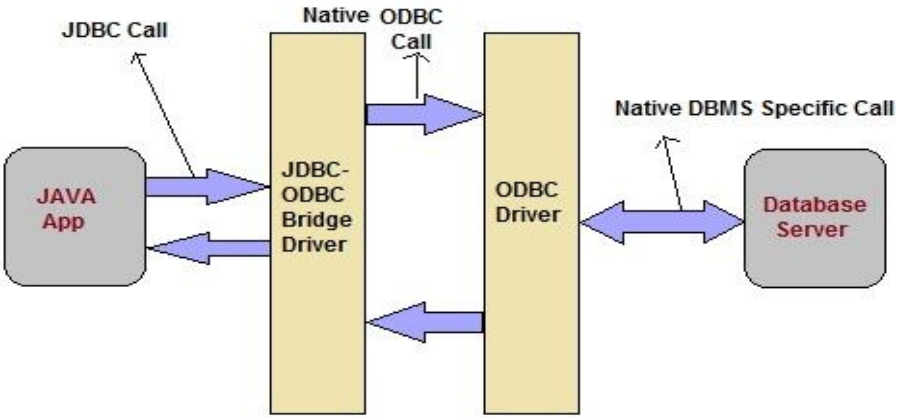

Fig 2: JDBC-ODBC Architecture

2.2 Type 2 or Native-API Driver

This type of driver make use of Java Native Interface(JNI) call on database specific native client API. These native client API are usually written in C and C++.

The type 2 drivers are two-tier drivers that have to execute on a client's machine. This type is faster as compared to JDBC-ODBC driver. This type requires native library and increased cost of application also.Figure 3 explains the Native API architecture.

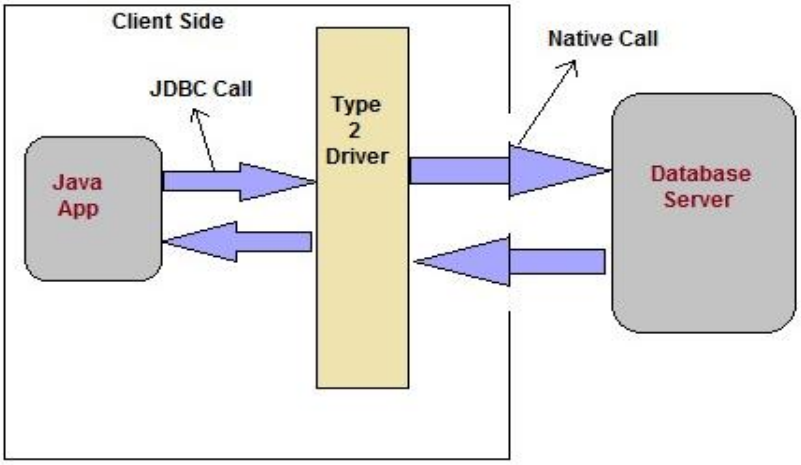

Fig 3: Native-API Driver Architecture

2.3 Type 3 or Network Protocol Driver

This driver translate the JDBC calls into a database server independent and Middleware server-specific calls. Middleware server further translate JDBC calls into database specific calls. The major advantage of the type 3 drivers is that the drivers do not need to be installed on the client machine. Further, they reduce the complexity as requests are passed through the network to the middle-tier server, which then translates the requests into the databasespecific native-connectivity interface for the database server. This type also provides the facility to switch over one database to another database. Figure 4 explains the Network Protocol Driver architecture.

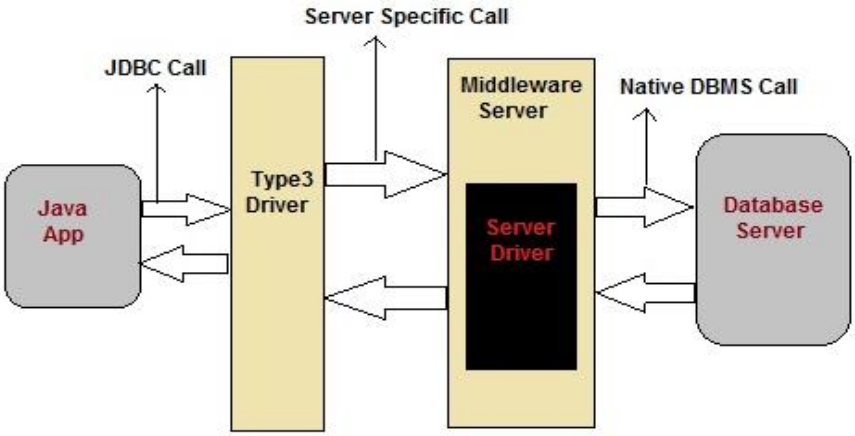

Fig 4: Netwok Protocol Driver Architecture

2.4 Type 4 or Thin Driver:

This is Driver called Pure Java Driver because. This driver interact directly with database. It does not require any native database library, that is why it is also known as Thin Driver. The Unity JDBC driver is of type 4, since it access databases indirectly through their corresponding JDBC drivers, and does not use a middleware server to connect to the individual databases. However, the Unity JDBC driver will use the JDBC drivers for each underlying database which may be of type 4. Note that since the Unity JDBC driver does not use a middleware server, all integration of results is performed on the client machine. Figure 5 explains the Thin Driver architecture.

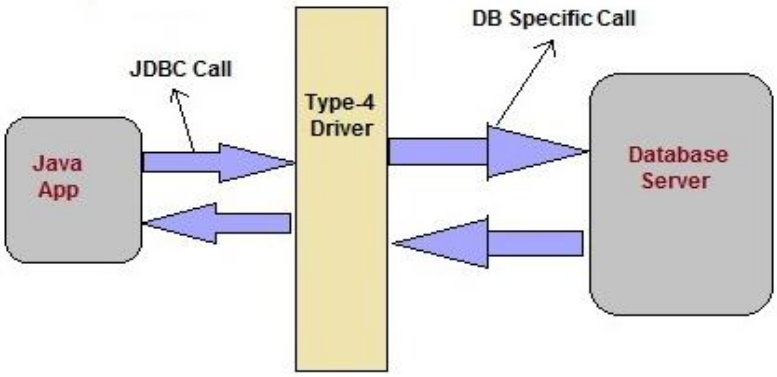

Fig 5: Thin Driver Architecture

III. COMPARISON OF JDBC DRIVERS

Table 1: Comparison of the 4 Types of JDBC Drivers

JDBC Drivers	**Pros**	**Cons**
type 1: JDBC-ODBC bridge drivers	integrated into JDK v1.1	1. Requires ODBC manager on the client; 2. The ODBC driver on each of the clients needs to be configured.
Type 2: Native API Partially-JAVA drivers	allows the programmer to fully utilize the speed and power that comes from the use of the API specifically developed for the DBMS	1. Driver is DBMS-dependent; 2. Requires a vendor-supplied DLL (Dynamic Link Library) to be installed in the client's JAVA library path.
Type 3: Net-Protocol All-JAVA drivers	DBMS-independent; allows the most flexible multi-server configuration	1. Requires a vendor-supplied intermediate server; 2. Configuration of the intermediate server
Type 4: Native Protocol All-JAVA drivers	Allows a direct call from the client to the database, without the need of client pre-configuration	1. Requires the use of protocols proprietary to the DBMS,; 2. Need to load a different driver for each DBMS that it needs to access.

IV. CONCLUSION

In this paper we have discussed different types of JDBC drivers under the context of a two-tier client/server model. However, it is entirely possible to use them to develop a multi-tier client/server application. The integration of web servers with database servers via the use of JAVA applets and JDBC drivers is useful for the teaching of database programming and web-based application development. The applet that we have developed, along with our experience of configuring the JDBC and JAVA environment, was used in a database course. Students built more complicated database/web applications on top of this sample applet.

REFERENCES

[1] Adam Rauch, LabKey Corporation, Seattle, WA "The Best of Both Worlds: Integrating a Java Web Application with SAS® Using the SAS/SHARE® Driver for JDBC",2010.

[2] Stavros Papastavrou & Panos K. Chrysanthis, George Samaras, Evaggelia Pitoura " An Evaluation of the Java-based Approaches to Web Database Access", 2002.

[3] Visibroker GateKeeper for Java: Programmer's Guide, Version 3.0. Borland, .

[4] Sun Microsystems Inc., JDBC drivers, http://www.oracle.com/technetwork/java/javase/jdbc/index.html.

[5] S. White, M. Fisher, R. Cattel, G. Hamilton and M. Hapner – JDBC API Tutorial and Reference, Second Edition – AddisonWesley, ISBN 0-201-43328-1, november 2001.

[6] Monika Jatiwal, Chetna Arora, Charu Arora, "A Study on advance java with database management",IJRDO, 2016.

