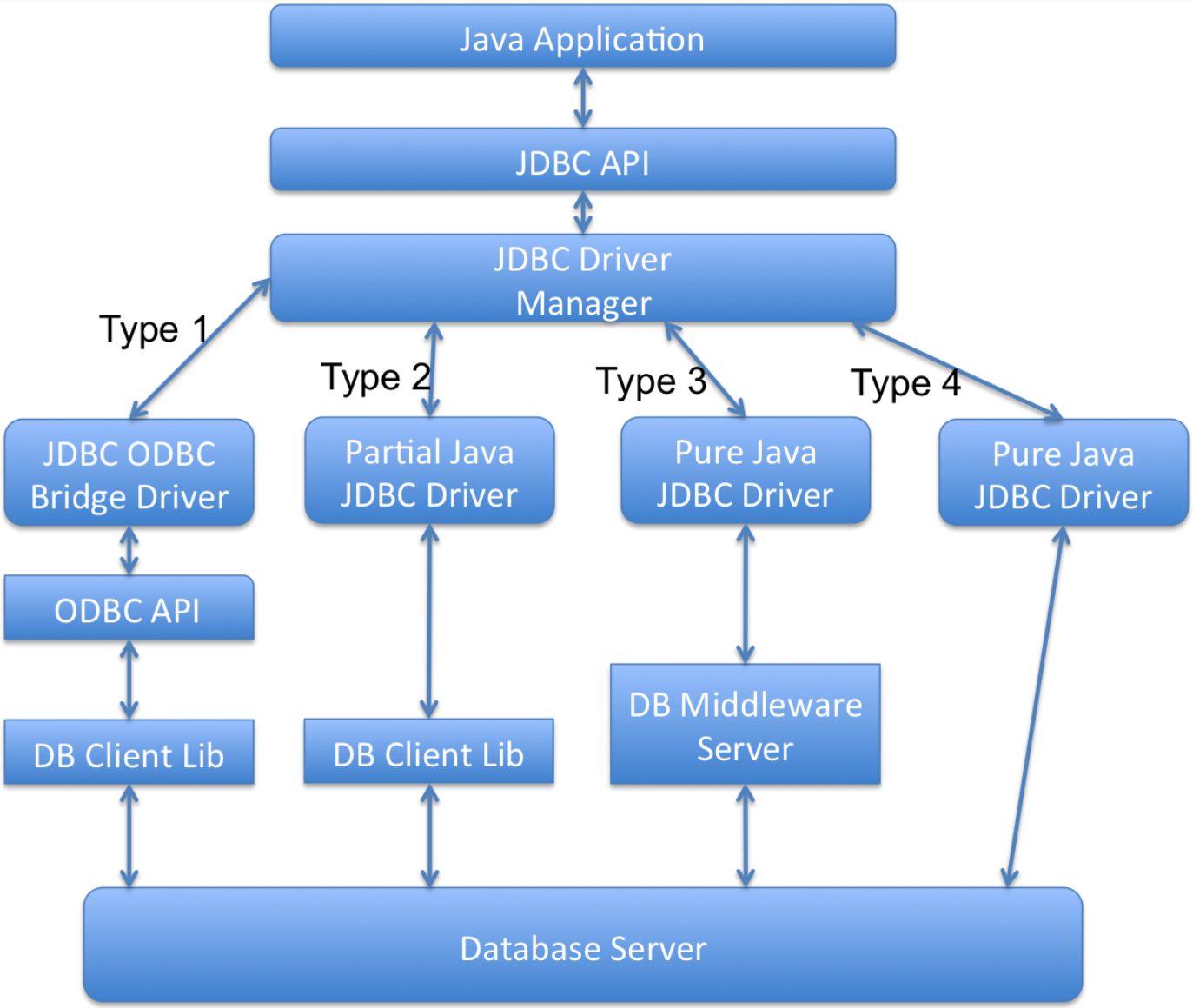
Introduction JDBC

Bok, Jong Soon
javaexpert@nate.com
https://github.com/swacademy

What is the JDBC™

- **Stands for Java™ Database Connectivity.**
- **Is an API (included in both J2SE and J2EE releases)**
- **Provides**
  - Cross-DBMS connectivity to a wide range of SQL database
  - Access to other tabular data sources such as spreadsheets or flat files.
- **The JDBC™ API provides universal data access from the Java™ programming language.**
- **Sun drew upon the successful aspects of one such API, ODBC.**

JDBC Architecture



Open Data Base Connectivity (ODBC)

- Was developed to create a single standard for database access in the Windows environment.
- Does not translate well into the Java world.
- Is a C API that requires intermediate APIs for other languages.

The JDBC API versus ODBC

- **ODBC is not appropriate for direct use.**
  - Calls from Java to native C code have a number of drawbacks in the security, implementation, robustness, and automatic portability of applications.
- **A literal translation of the ODBC C API into a Java API would not be desirable.**
  - Java has not pointers, and ODBC makes copious use of them.
- **ODBC is hard to learn.**
  - It mixes simple and advanced features together, and it has complex options even for simple queries.
  - A Java programmer using JDBC does not need to worry about either memory management or data byte alignment.

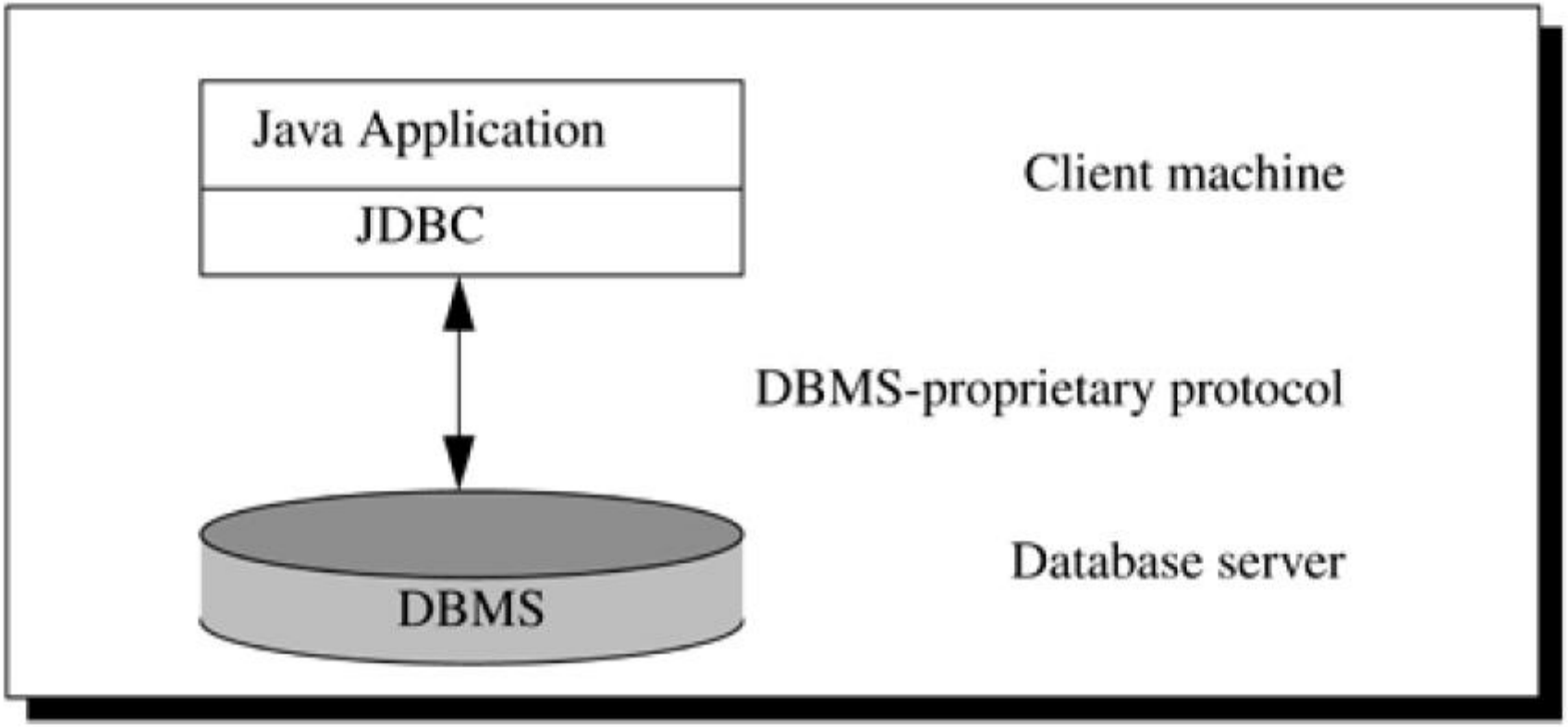
The JDBC API Packages

- Provides programmatic access to relational data from the Java programming language.
- Can be used to interact with multiple data sources in a distributed, heterogeneous environment.
- Introduce in January 1997.
- java.sql
- javax.sql
- Automatically get both packages when you download the Java™ 2 Platform, Standard Edition, Version 7 (J2SE™)

The JDBC API versus ODBC (Cont.)

- **A Java API like JDBC is needed in order to enable a "pure Java" solution.**
  - When ODBC is used, the ODBC driver manager and drivers must be manually installed on every client machine.
  - When the JDBC driver is written completely in Java, JDBC code is automatically installable, portable, and secure on all Java platforms, from network computers to mainframes.
- **The JDBC 3.0 API includes functionality that is not available with ODBC.**
  - ODBC does not support SQL99 data types, auto-generated keys, or savepoints.

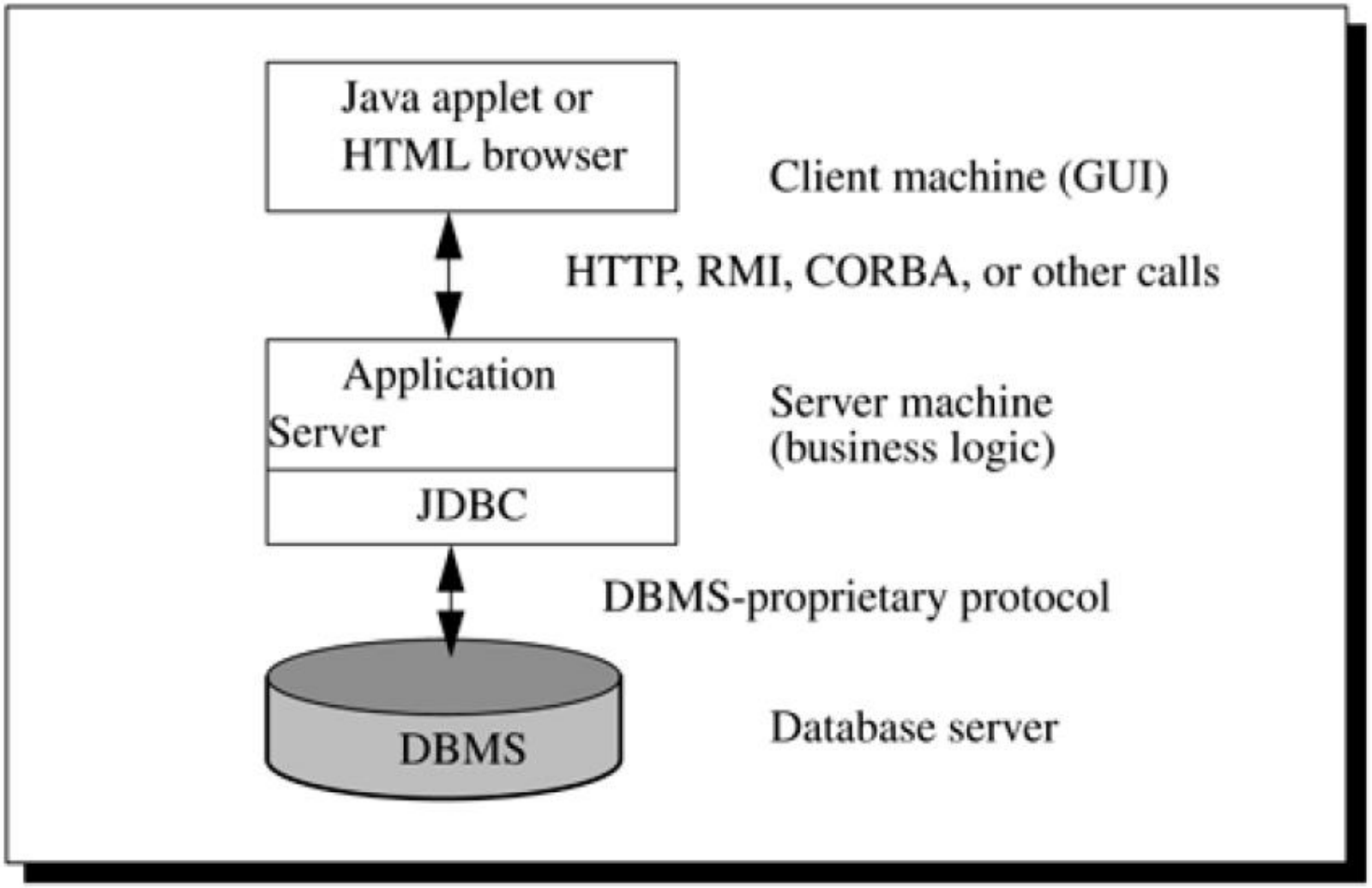
Two-tier Model



Two-tier Model (Cont.)

- Talks directly to the data source.
- Requires a JDBC driver that can communicate with the particular data source being accessed.
- The data source may be located on another machine to which the user is connected via a network.
- Is referred to as a client/server configuration.

Three-tier Model



Three-tier Model (Cont.)

- Commands are sent to a “middle tier” of services, which then sends the commands to the data source.
- The data source processes the commands and sends the results back to the middle tier, which then sends them to the user.
- The middle tier makes it possible to maintain control over access and the kinds of updates that can be corporate data.
- Another advantage is that it simplifies the deployment of applications.

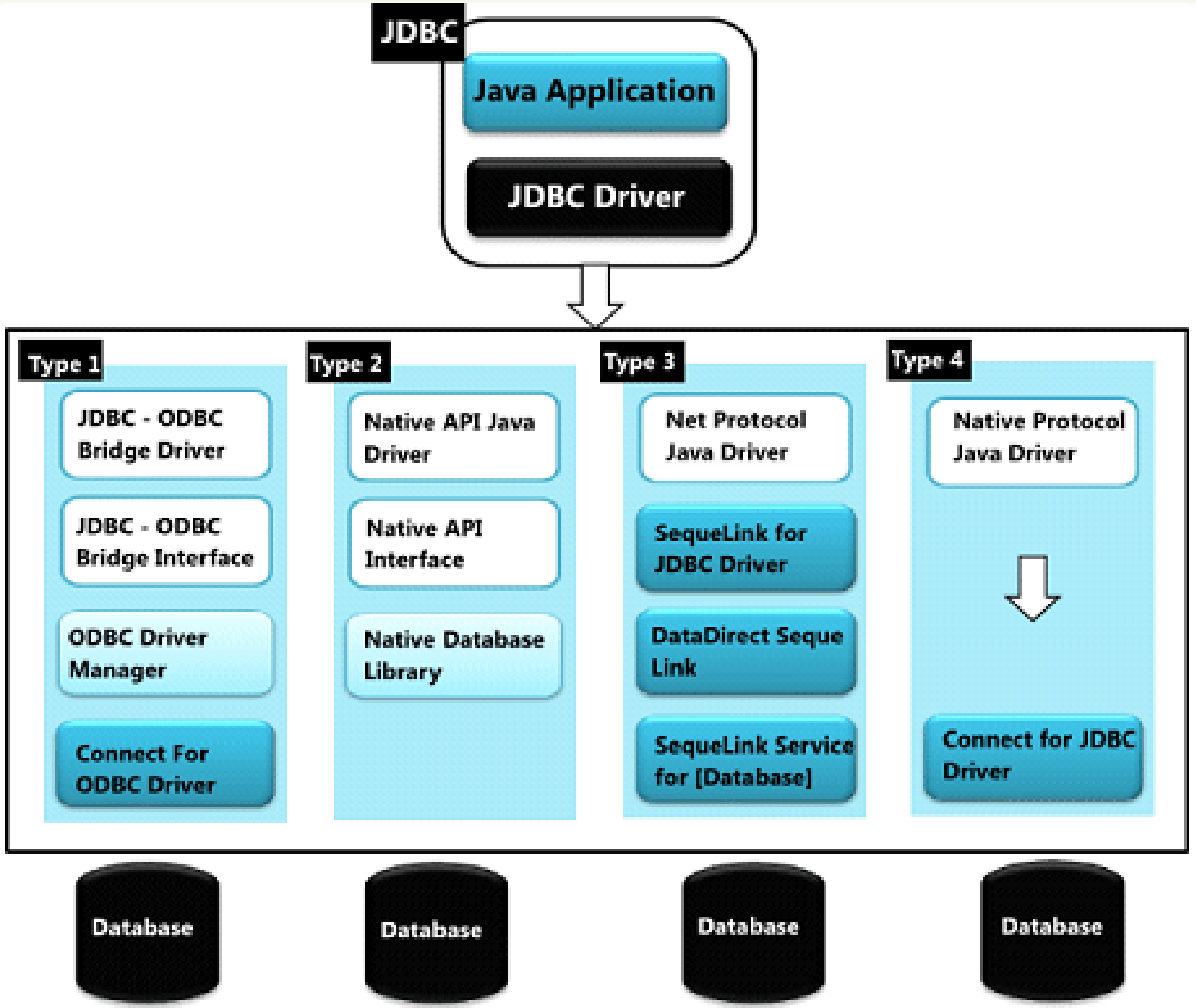
SQL Conformance

- **Problems : SQL is not yet as standard as one would like.**
  - Data types used by different DBMSs sometimes vary, and the variations can be significant.
  - Although most DBMSs use a standard form of SQL for basic functionality, they do not conform to the more recently defined standard SQL syntax or semantics for more advanced functionality.
  - Not all databases support stored procedures or outer joins, and those that do are not always consistent with each other.
  - Support for SQL99 features and data types varies greatly.

SQL Conformance (Cont.)

- **Solutions with JDBC :**
  - To define a set of generic SQL type identifiers in the class `java.sql.Types`.
  - To allow any query string to be passed through to an underlying DBMS driver.
  - To provide ODBC-style escape clauses.
  - Provides descriptive information about the DBMS by means of the interface `DatabaseMetaData` so that applications can adapt to the requirements and capabilities of each DBMS.
  - A JDBC driver must support at least ANSI SQL92 Entry Level.

JDBC Driver Types

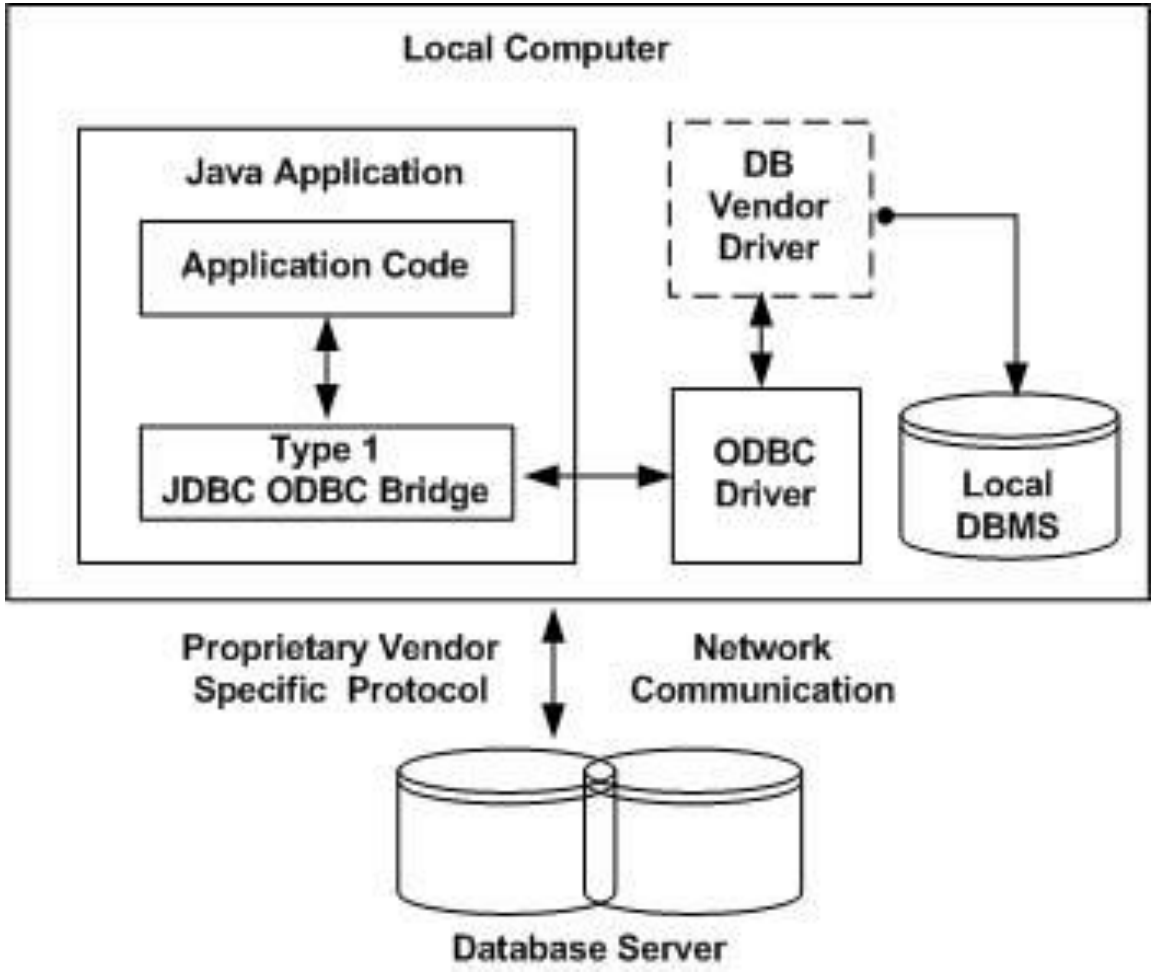


JDBC Driver Types (Cont.)

- **JDBC-ODBC bridge driver plus ODBC driver**
  - The Sun Microsystems bridge product provides JDBC access via ODBC drivers.
  - Note that ODBC binary code, and in many cases database client code, must be loaded on each client machine that uses this driver.
  - Using ODBC requires configuring on your system a Data Source Name (DSN) that represents the target database.
  - When Java first came out, this was a useful driver because most databases only supported ODBC access.

JDBC Driver Types (Cont.)

- JDBC-ODBC bridge driver plus ODBC driver

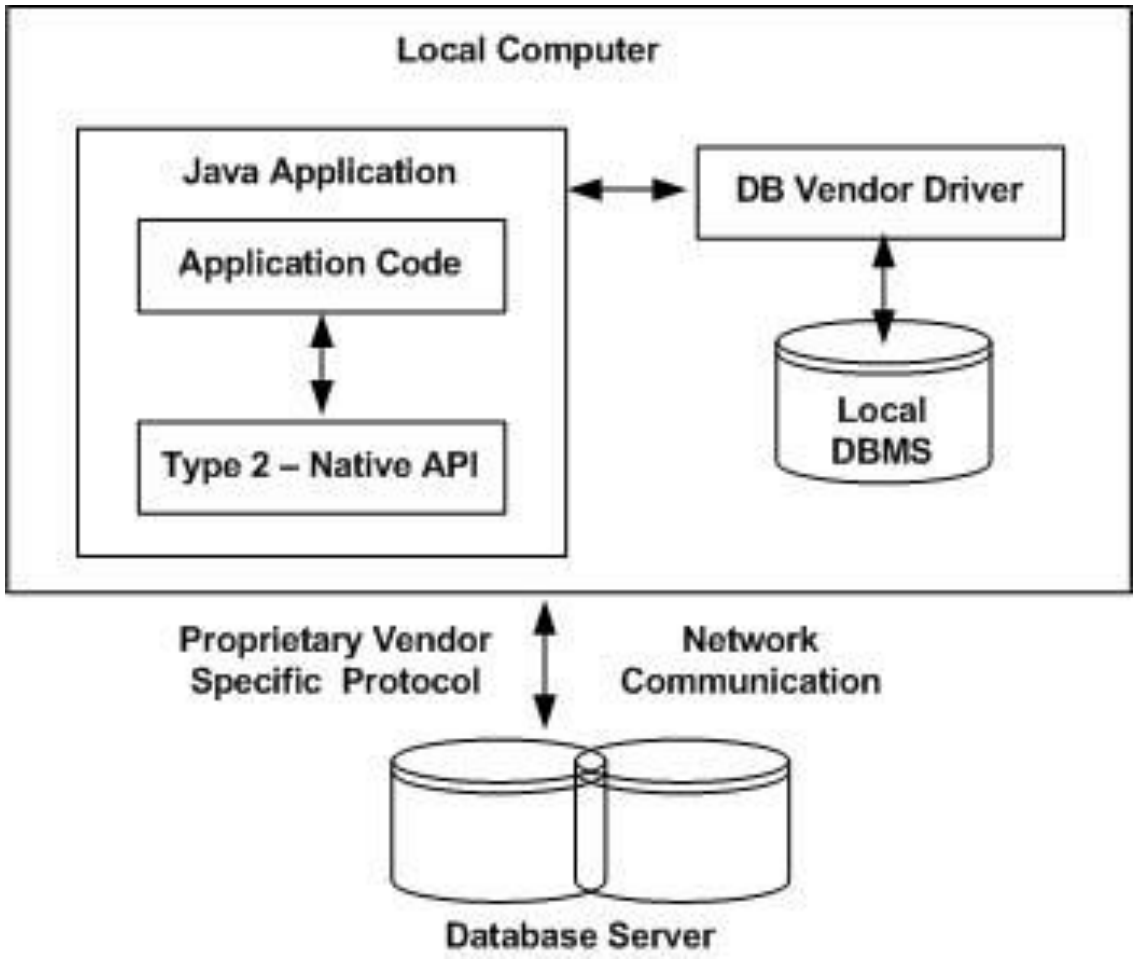


JDBC Driver Types (Cont.)

- **Native-API partly Java driver**
  - This kind of driver converts JDBC calls into calls on the client API for Oracle, Sybase, Informix, IBM DB2, or other DBMSs.
  - Note that, like the bridge driver, this style of driver requires that some operating system-specific binary code be loaded on each client machine.
  - Calls are converted into native C/C++ API calls which are unique to the database.
  - These drivers typically provided by the database vendo rs and used in the same manner as the JDBC-ODBC B ridge, the vendor-specific driver must be installed on e ach client machine.

JDBC Driver Types (Cont.)

- **Native-API partly Java driver**

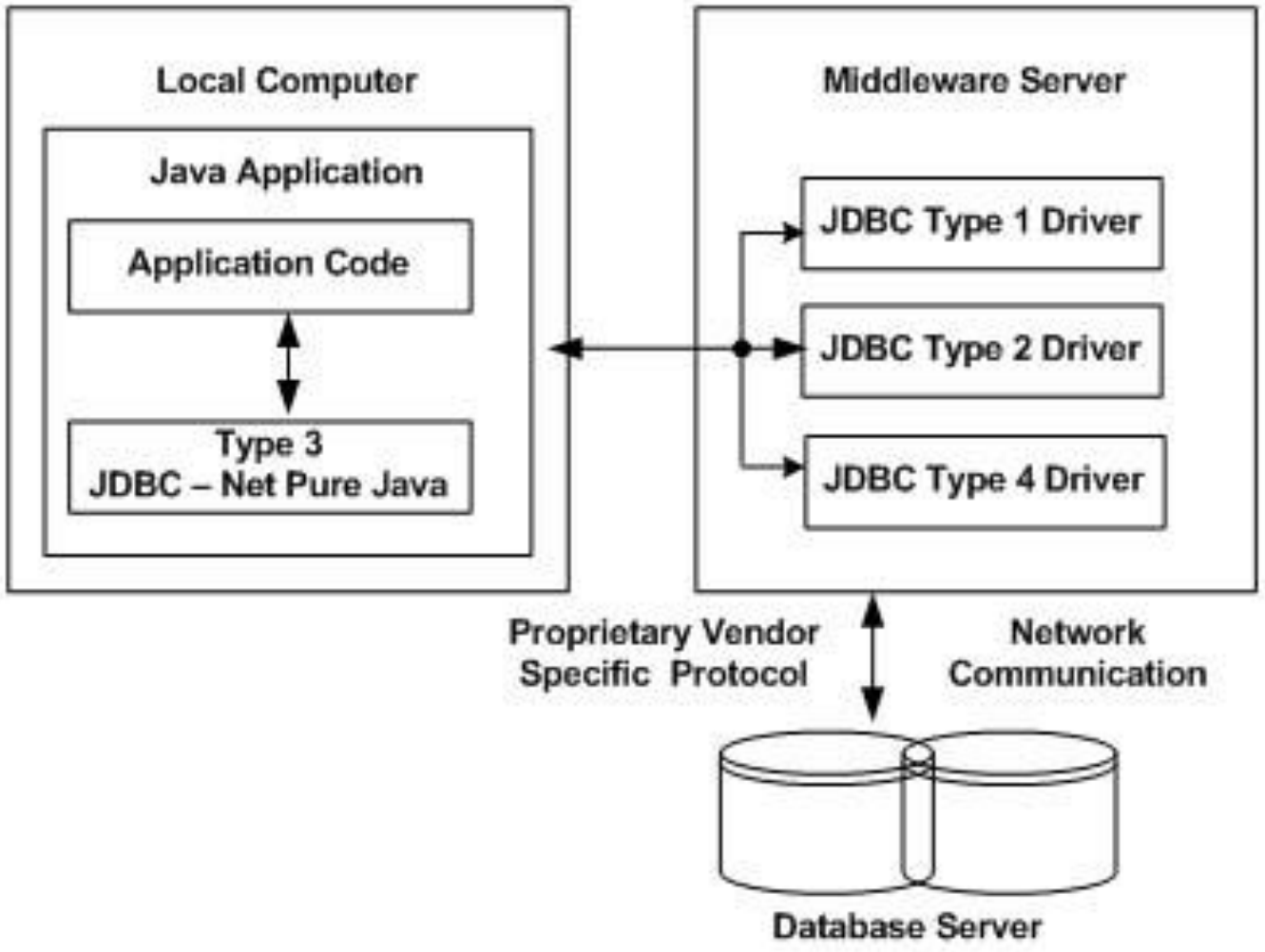


JDBC Driver Types (Cont.)

- **JDBC-Net pure Java driver**
  - A three-tier approach is used to accessing databases.
  - The JDBC clients use standard network sockets to communicate with an middleware application server.
  - The socket information is then translated by the middleware application server into the call format required by the DBMS, and forwarded to the database server.
  - This kind of driver is extremely flexible, since it requires no code installed on the client and a single driver can actually provide access to multiple databases.

JDBC Driver Types (Cont.)

- JDBC-Net pure Java driver

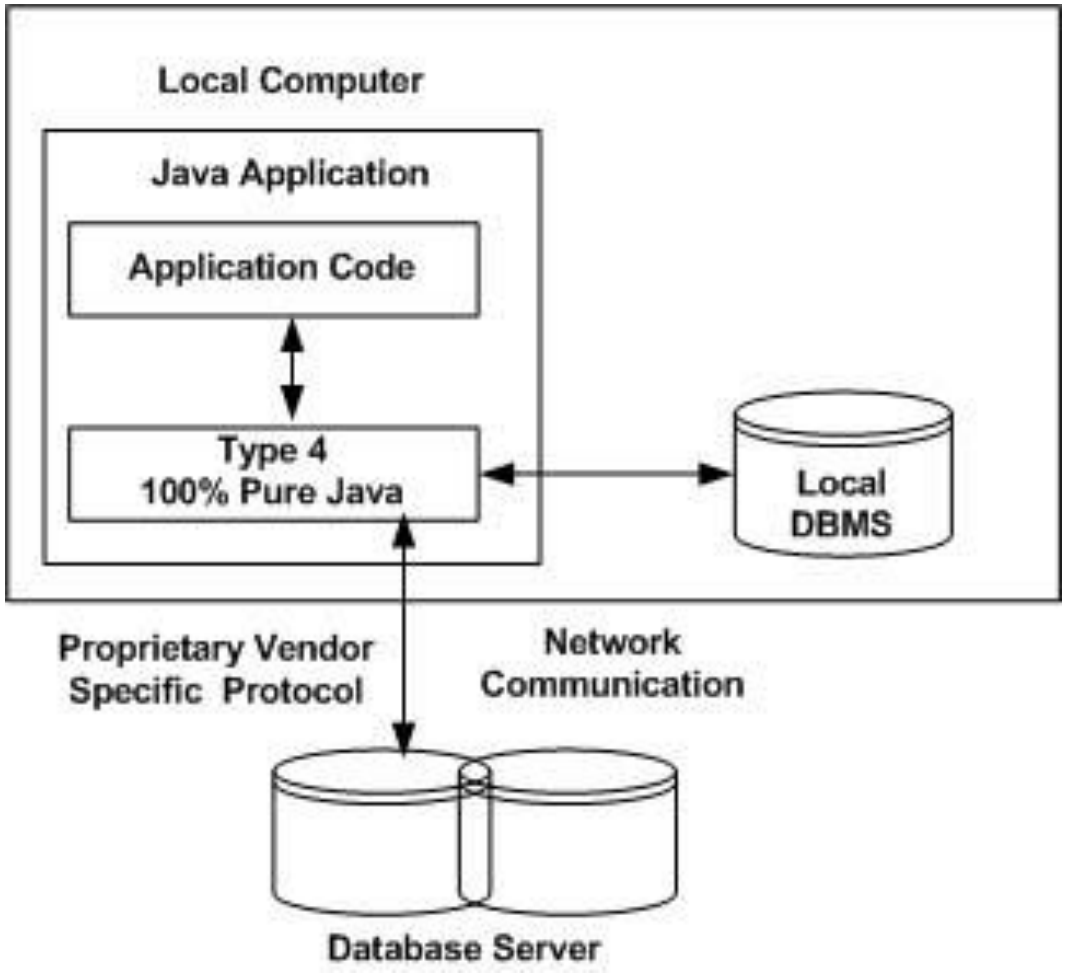


JDBC Driver Types (Cont.)

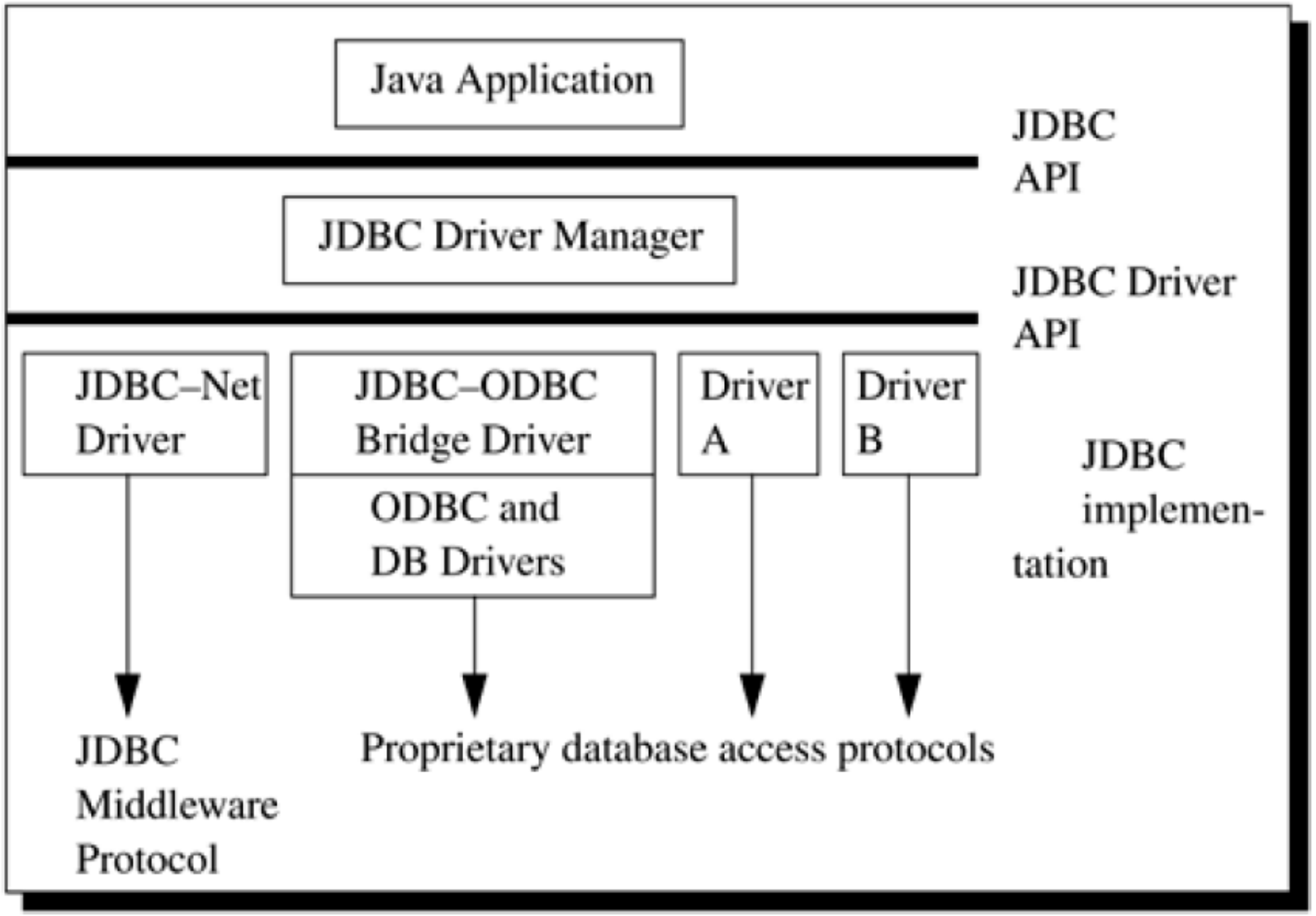
- **Native-protocol pure Java driver**
  - This kind of driver converts JDBC calls directly into the network protocol used by DBMSs.
  - This allows a direct call from the client machine to the DBMS server and is an excellent solution for intranet access.
  - Several that are now available include Oracle, Sybase, IBM DB2, Borland InterBase, and Microsoft SQL Server.

JDBC Driver Types (Cont.)

- **Native-protocol pure Java driver**



JDBC Driver Types (Cont.)



Which Driver Should be Used?

- If you are accessing one type of database, such as Oracle, Sybase, or IBM, the preferred driver type is 4.
- If your Java application is accessing multiple types of databases at the same time, type 3 is the preferred driver.
- Type 2 drivers are useful in situations where a type 3 or type 4 driver is not available yet for your database.
- The type 1 driver is not considered a deployment-level driver and is typically used for development and testing purposes only.

JDBC Driver Types (Cont.)

Driver Category	All Java	Network Connection
JDBC-ODBC Bridge	No	Direct
Native API as basic	No	Direct
JDBC-Net	Client, Yes server, Maybe	Indirect
Native protocol as basic	Yes	Direct

Related Resources

- **JDBC™ Home**
  - http://www.oracle.com/technetwork/java/javase/jdbc/index.html
- **Java DB Home**
  - http://docs.oracle.com/javadb/index_jdk8.html
- **JDBC™ Database Access Tutorial**
  - http://docs.oracle.com/javase/tutorial/jdbc/