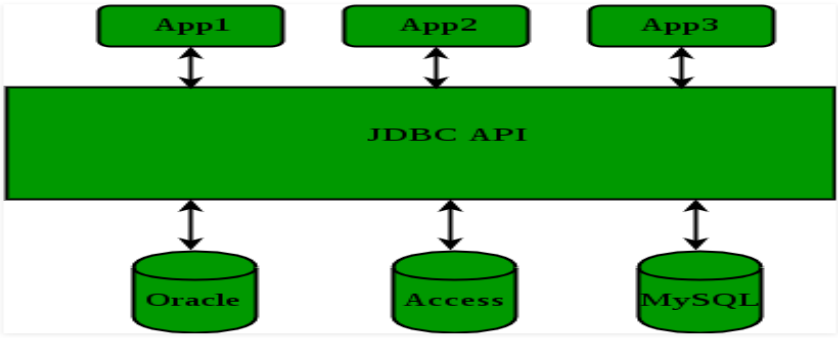
JDBC Drivers

- **Need for Drivers:** Application programs are not directly to interface with the database. (convert programming language requests to database queries)
- **Solution:** API (JDBC Drivers)
- **Purpose of JDBC**:
  - **Interaction with Database**: JDBC Driver is a software component that enables java application to interact with the database. (accessing of tabular data)
  - **Three tier Architecture (Middle Layer)**: It acts as a middle layer interface between java applications and database.
- **Structure of Java API**:



JDBC Drivers

- **Types of JDBC drivers (4 types)**:
  1) Type-1 driver or JDBC-ODBC bridge driver
  2) Type-2 driver or Native-API driver (partially java driver)
  3) Type-3 driver or Network Protocol driver (fully java driver) (middleware)
  4) Type-4 driver or Thin driver (fully java driver)

1. Type-1 Driver or JDBC – ODBC driver:

- **Driver name** : ODBC driver (sun.jdbc.odbc.JdbcOdbcDriver)
- **Alias Name** : Universal Server – because it is used to connect with any database (database independent server)
- **Purpose**: It converts Java interface function calls into ODBC function calls using ODBC driver.
- **Security** : No
- **Installation of drivers**: Required in client machine (built-in with JDK)
- **Portable:** No. (not written in java)
- **Need of DSN (Data Source Name) :** Yes

- **Removing Version:** From Java 8
- **Platform Dependent :** Implemented in C Language
- **Representation of Type-1 Driver**:

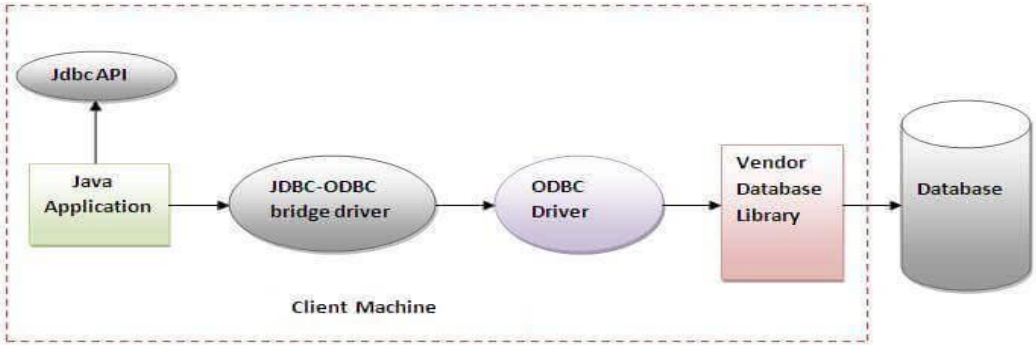


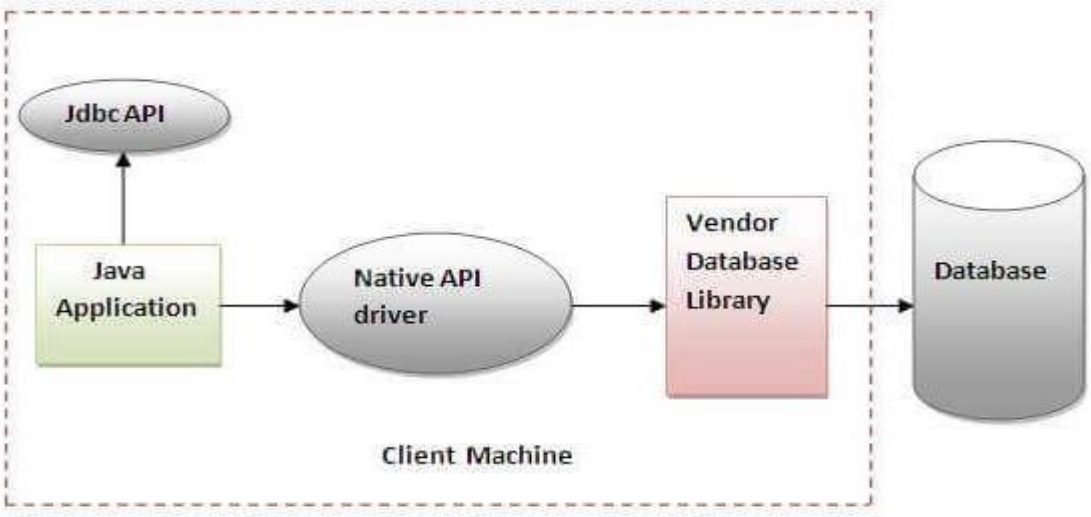
Figure- JDBC-ODBC Bridge Driver

- **Advantages**:
  i. Easy to use
  ii. Suggested for experimental purpose
  iii. Easy to connect with all the database
- **Disadvantages**:
  i. Performance degradation - because of conversion
  ii. Installation required
  iii. Not suitable for applet (because of install drivers in client machine)
  iv. Specific ODBC drivers not available in all platform
  v. No support from JDK 1.8

2. Type -2 or Native API Driver

- **Drivers**: Supported (provided) by client vendors
- **Purpose**: Convert JDBC method calls into native calls (C/C++ APPI) of the database API.
- **Installation:** Installation of libraries in client side machine is needed
- **Portable**: No (Not entirely written in java)
- **Security** : Partial
- **Database independent** : No (written in C language)
- **Example Driver**: Oracle Call Interface (OCI) [oracle provides native API and native protocol. Whereas MySQL has given only native protocol]

- **Native API Driver**:

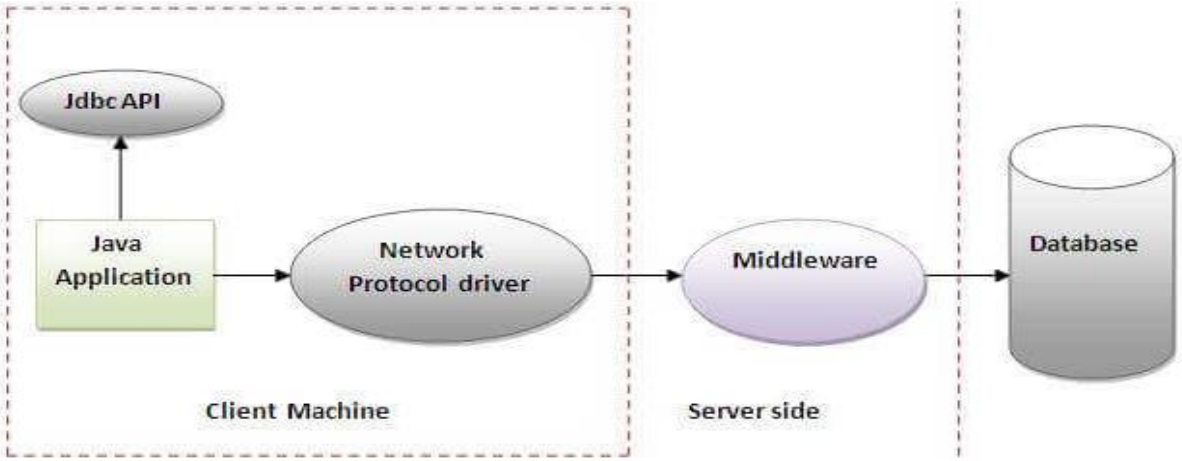


- **Advantages**:
    i. Performance upgraded than type 1 driver.
    ii. No implementation of JDBC-ODBC bridge
- **Disadvantages**:
    i. Driver need to installed in client machine
    ii. Vendor client library need to installed on client machine
    iii. Platform dependent

3. Type-3 or Network Protocol Driver: (Middleware)

- **Drivers**: Supported (provided) by client vendors
- **Purpose**: Convert JDBC method calls into directly or indirectly into the vendor-specific database protocol. Middleware server convert into database calls
- **Installation**: Installation not required in client machine.
- **Portable**: Yes (written in java)
- **Security** : yes
- **Database independent** : yes (platform independent)
- **Network support** : Needed
- **Size of drivers**: small and load quickly
- **Multiple database:** Single driver is enough to connect with multiple database
- **Example :** IDA server, D Edwards EnterpriseOne Data Access Server (DAS)

- **Structure of Type-3 Driver**:

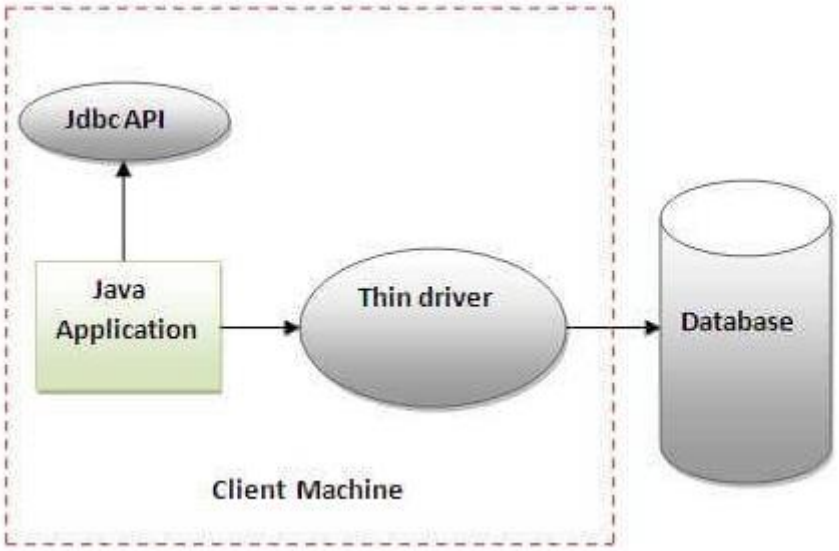


- **Advantages**:
  i. No installation required in client side machine.
  ii. No need for vendor-based database library to be present in client machine
  iii. It supports for load balancing, scalability, caching etc..,
  iv. Flexible to access the multiple databases by using single driver
- **Disadvantages**:
  i. Network support is required on client machine.
  ii. Requires database-specific coding to be done in the middle tier.
  iii. Cost is high because of middleware implementation and maintenance
  iv. Need to implement database specific coding in middleware.
  v. Complexity is high

4. Type 4 or Thin Driver:

- **Drivers**: Supported (provided) by network libraries which are directly communicate with server through socket connection
- **Automatic downloaded dynamically**
- **Purpose**: converts JDBC calls directly into the vendor-specific database protocol. Direct interaction with the database
- **Installation**: Installation is not required in client and server machine. (install inside of JVM)
- **Portable**: Yes (written in java)
- **Security** : yes
- **Database independent** : yes (platform independent)
- **Network support** : Needed

- **Size of drivers**: small and load quickly
- **Multiple database:** Single driver is enough to connect with multiple database
- **Example:** Oracle, SYBASE, IBM database preferred to use thin4 driver.
- **Example Drivers**: MySQL's Connector/J driver



- **Advantages**:
  i. It is platform independent means pure Java driver.
  ii. Performance is very good in comparison to all other drivers.
  iii. No need of any middleware services.
  iv. All the processes of application are managed by JVM.
  v. No need for conversion of intermediate formats.
- **Disadvantages**:
  i. Need a separate driver for each database.