JDBC

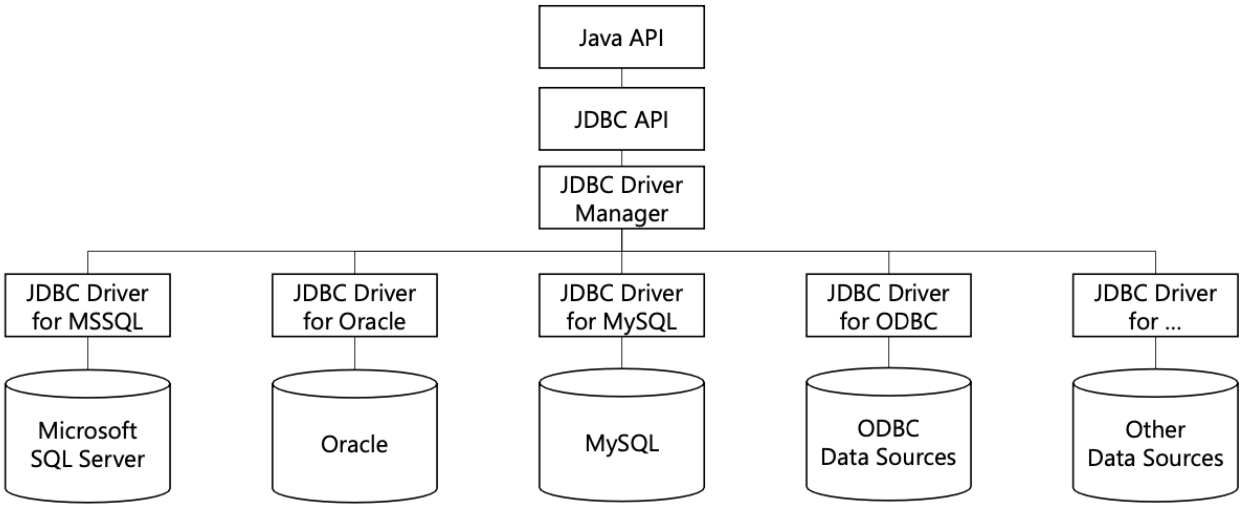
JDBC 소개

JDBC

- 관계형 데이터베이스에 저장된 데이터를 접근 및 조작 할 수 있게 하는 java api
  - Java 응용 프로그램이 다양한 DBMS에 대해 일관된 API로 데이터베이스에 연결, 검색, 수정, 관리
  - DBMS의 종류에 관계없이 동일한 방법으로 데이터베이스에 접근

JDBC 구조

- 네트워크상의 데이터베이스에 접속할 수 있도록 하는 Database 연결 기능 제공
- JDBC API, JDBC Driver Manager, JDBC Driver로 구성



구성요소	설명	역할
Java Application	Java 응용 프로그램 및 Java 웹 응용프로그램 서버 (Tomcat, Weblogic 등)	응용 프로그램 개발자, 웹 응용프로그램 서버개발사
JDBC Driver Manager	JDBC 드라이버 로드	Java SE 개발사
JDBC API	데이터베이스 연결 및 제어를 위한 인터페이스/클래스	Java SE 개발사
JDBC Driver	데이터베이스 개발사에서 만든 데이터베이스 드라이버	데이터베이스 개발사

JDBC 표준

다운로드

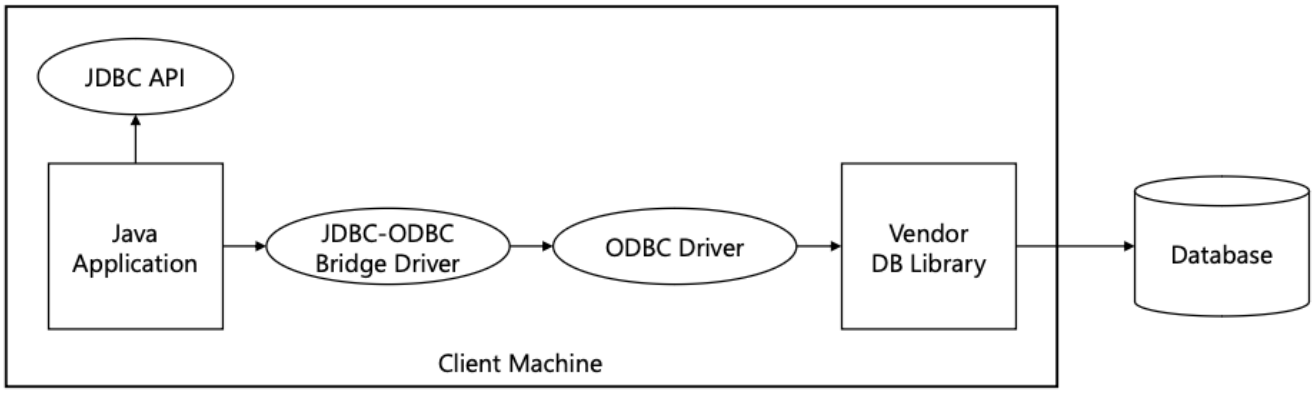
- JDBC는 Java 프로그래밍 언어와 다양한 데이터베이스의 독립적 연결을 위한 표준
- JDBC API는 SQL 기반 데이터베이스 액세스 API를 제공
- JDBC 기술을 사용하면 Java 프로그래밍 언어를 사용해 엔터프라이즈 데이터에 엑세스 해야하는 응용프로그램에 대해 WORE 기능을 활용할 수 있음.

JDBC Driver Type

- JDBC-ODBC Bridge Type
- Native_API / Partly Java Type
- Net-Protocol / All-Java Type
- Native-Protocol / All-Java Type

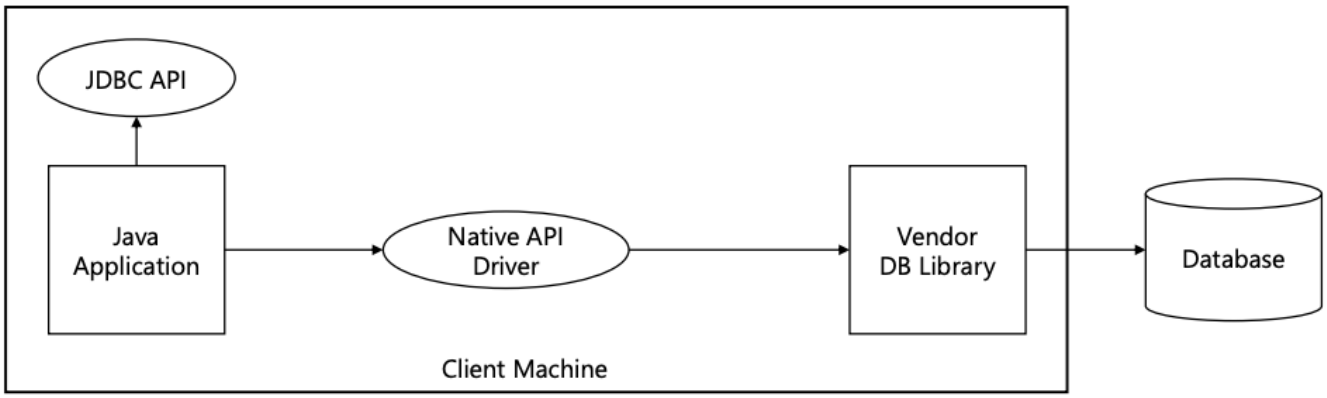
Type 1: JDBC-ODBC Bridge

- JDBC와 ODBC 드라이버 간의 다리 역할 수행
    - ODBC(Open Database Connectivity)
    - Microsoft가 만든 DBMS 연결 표준
- JDBC 호출을 ODBC 호출로 변환하고 요청을 ODBC 드라이버에 전송
- 설정이 간편하지만 실행 속도가 느림



Type 2: Native API / Partly java

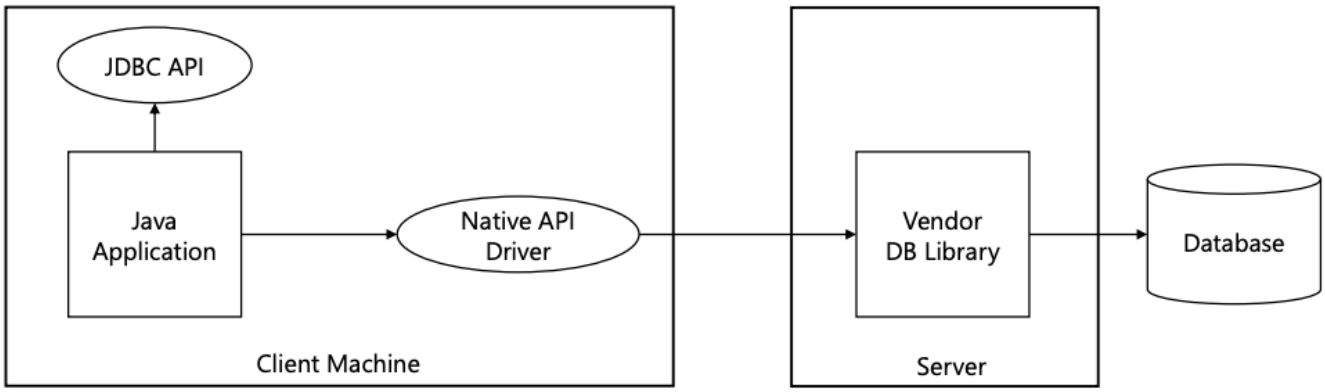
- JNI(Java Native interface)를 사용해 데이터 베이스전용 클라이언트 호출
- 타입1 보다 빠르지만 NativeLibrary를 설치 해야함
- 데이터베이스에 종속적



Type 3: Net Protocol / All Java

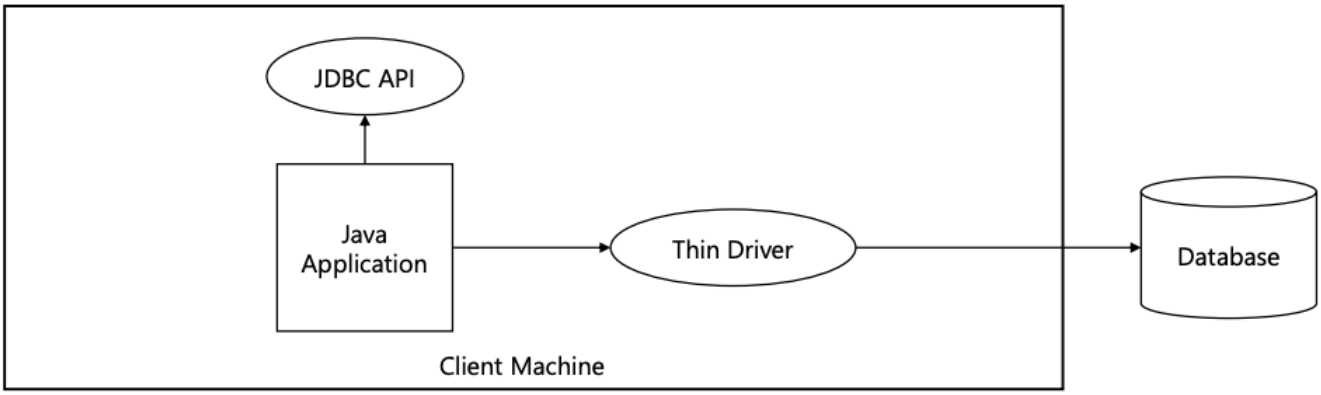
- JDBC 미들웨어 서버와 독점프로토콜로 통신
- JDBC 미들웨어가 요청된 프로토콜을 데이터베이스 호출로 변환
- 데이터베이스에 독립적

- 많은 네트워크 호출로 인해 상대적으로 느림



Type 4: Native-Protocol / All-Java

- 직접 데이터베이스와 통신, 100% 순수 자바 구현
- 가장 많이 사용되는 형태
- 별도의 미들웨어 또는 네이티브 라이브러리가 필요하지 않음
- 빠른 실행 속도와 플랫폼 독립적
- 가장 좋은 성능



JDBC 아키텍처

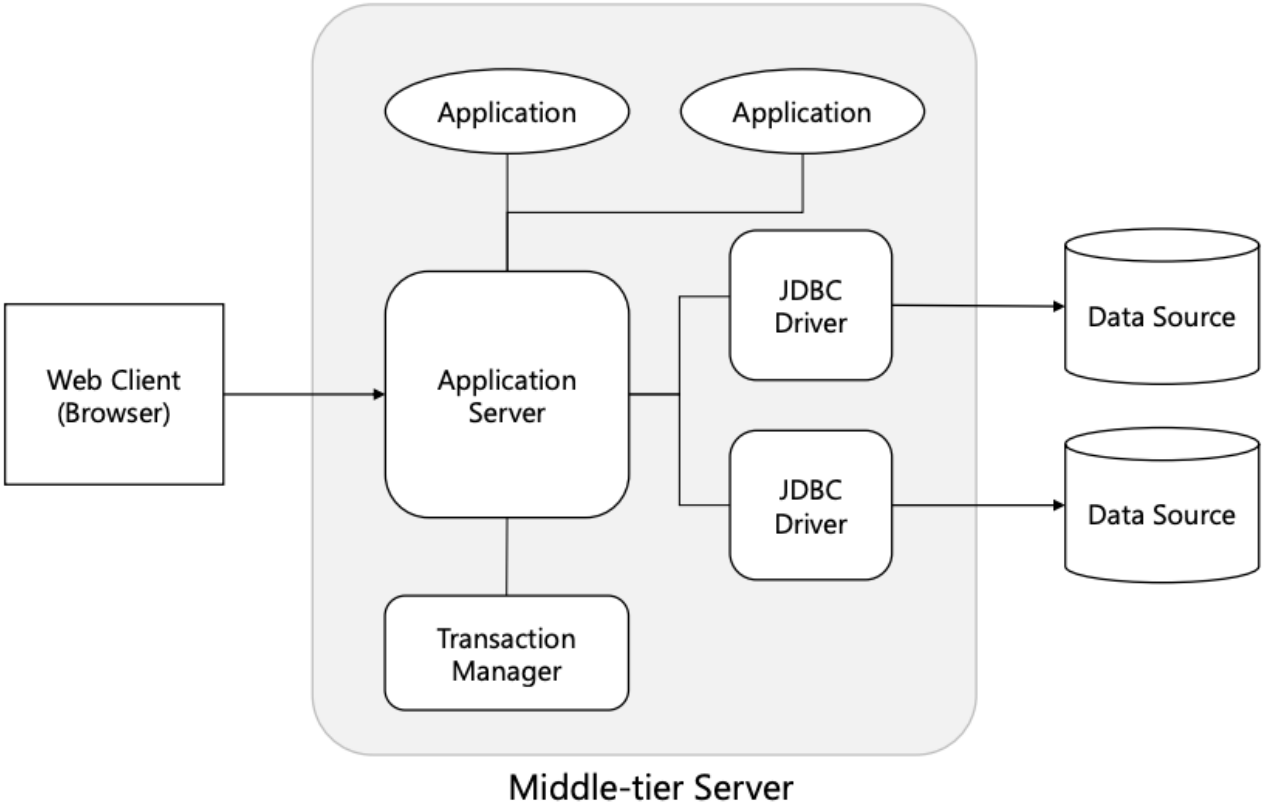
2-TIER 아키텍처

- Client Layer와 Server Layer로 구성
- 자바 응용프로그램이 데이터베이스와 직접 통신하는데 도움이 됨
- JDBC 드라이버를 사용하여 특정 데이터베이스와 통신
- 쿼리 또는 요청을 사용자가 데이터베이스로 전송하고 결과를 사용자에 의해 수신
- 데이터베이스는 동일한 컴퓨터 또는 네트워크 상의 원격 시스템이 있을 수 있음
- 클라이언트 프로그램이 바로 데이터베이스에 접속하는 구성- 확장성에 제한
- Client-Server 아키텍처(또는 구성)

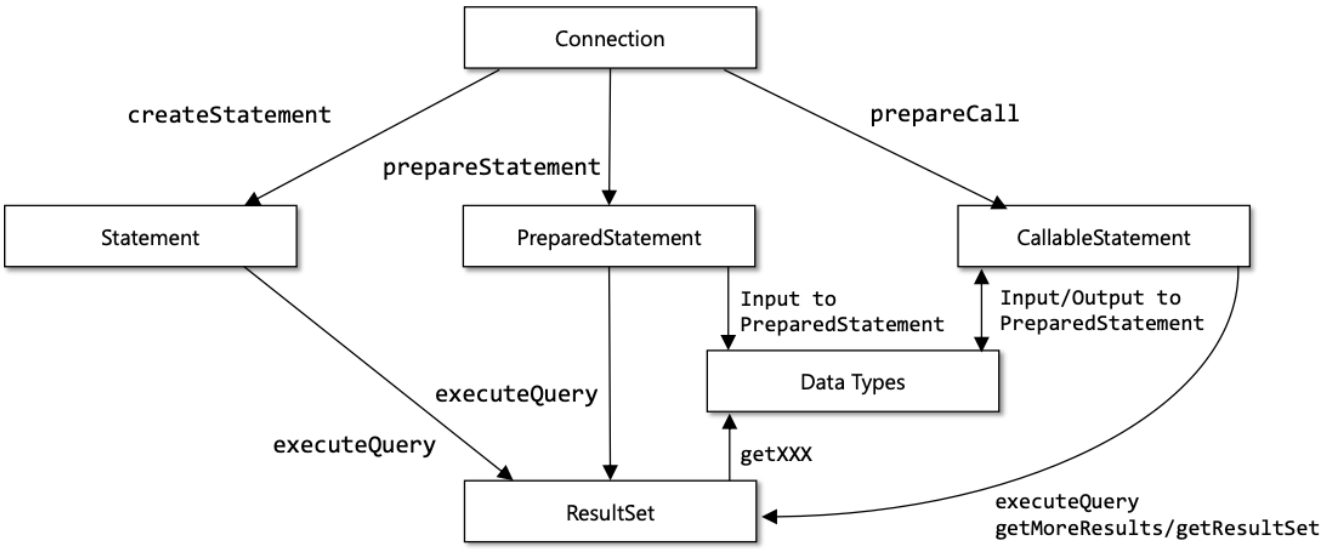
3-TIER 아키텍처

- Business Logic을 포함하는 미들웨어가 추가됨
    - 클라이언트에 데이터베이스로의 직접적인 통신은 없음

- 사용자의 요청은 미들웨어로 전송됨
- 데이터베이스는 요청을 처리하고 결과를 중간 계층으로 보낸 다음 사용자와 통신
- 성능을 높이고 응용프로그램 배포를 단순화
- Client Tier
    - 사용자와 상호작용하는 화면들로만 구성된 Thin Layer
    - 자바 프로그램, 웹 브라우저, PDA, 스마트폰 등
- Middle Tier Server
    - 응용프로그램은 클라이언트와 상호작용하는 비즈니스 로직 구현체
    - 응용프로그램 서버는 다양한 응용시스템 구현을 위한 기반 기능 제공
    - 일반적으로 Tomcat과 같은 JAVA EE서버로, 서버에서 JDBC 드라이버를 직접적으로 사용



java.sql 패키지



여담

ODBC의 탄생 설화...

ODBC(Open Database Connectivity)는 C 언어에서 데이터베이스와 연결하여 데이터를 관리하기 위한 API(Application Programming Interface)입니다. ODBC의 탄생 설화는 다음과 같습니다.

1990년대 초반, 다양한 데이터베이스 시스템이 등장하면서 각각의 데이터베이스 시스템에 대응하는 별도의 API가 필요해졌습니다. 이러한 상황에서 Microsoft는 다양한 데이터베이스 시스템에 대응하는 표준 API를 만들고자 하였습니다. 그리고 1992년에 ODBC API를 발표하였습니다.

ODBC는 데이터베이스의 종류나 버전에 상관없이, 일관된 방법으로 데이터베이스와 연결하고 데이터를 관리할 수 있게끔 설계되었습니다. 이로 인해 C 언어를 사용하는 개발자들은 다양한 데이터베이스에 접속하고 데이터를 쉽게 다룰 수 있게 되었으며, ODBC는 현재까지도 C 언어에서 가장 많이 사용되는 데이터베이스 연결 API 중 하나입니다. ODBC는 자바에서 사용되는 JDBC API의 모태가 되기도 했습니다

JDBC의 탄생 설화

JDBC(Java Database Connectivity)는 자바에서 데이터베이스와 연결하여 데이터를 관리하기 위한 API(Application Programming Interface)입니다. JDBC의 탄생 설화는 다음과 같습니다.

1990년대 초반, 자바가 처음 출시되었을 때는 웹 애플리케이션을 만드는 것이 목적이었습니다. 그러나 당시 자바는 데이터베이스와의 연결이 불편했습니다. 이에 자바 개발자들은 C++에서 사용되던 ODBC(Open Database Connectivity)를 모방하여, 자바에서 데이터베이스와의 연결을 쉽게 하기 위한 API를 만들고자 하였습니다. 그래서 1996년 Sun Microsystems(현재 오라클)에서 JDBC API를 공식적으로 발표하였습니다.

JDBC API는 데이터베이스의 종류나 버전에 상관없이, 일관된 방법으로 데이터베이스와 연결하고 데이터를 관리할 수 있게끔 설계되었습니다. 이로 인해 자바 개발자들은 다양한 데이터베이스에 접속하고 데이터

를 쉽게 다룰 수 있게 되었으며, JDBC는 현재까지도 자바에서 가장 많이 사용되는 데이터베이스 연결 API 중 하나입니다.