seed
beyond the obvious

Core Java

Introduction to JDBC

Objectives

- State what is Java Database Connectivity
- State different types of drivers supported by JDBC
- Describe the steps to be followed for writing a simple JDBC application
- Describe the use of Resultset interface

seed
beyond the obvious

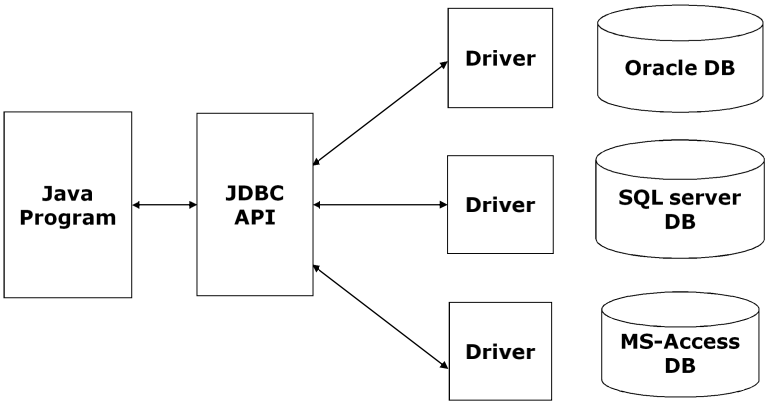
JDBC

- Lets programmers connect to a database, query it or update through a Java application.
- Programs developed with Java & JDBC are platform & vendor independent.
- JDBC library is implemented in java.sql package.

JDBC

- A driver is a program that converts the Java method calls to the corresponding method calls understandable by the database in use.

JDBC



seed
beyond the obvious

ODBC

- A driver manager for managing drivers for SQL based databases.
- Developed by Microsoft to allow generic access to disparate database systems on windows platform.
- J2SDK comes with JDBC-to-ODBC bridge database driver to allow a java program to access any ODBC data source.

seed
beyond the obvious

JDBC Vs ODBC

- ODBC is a 'C' API
- ODBC is hard to learn – because of low-level native ODBC.
- ODBC most suited for only Windows platform
- No platform independence

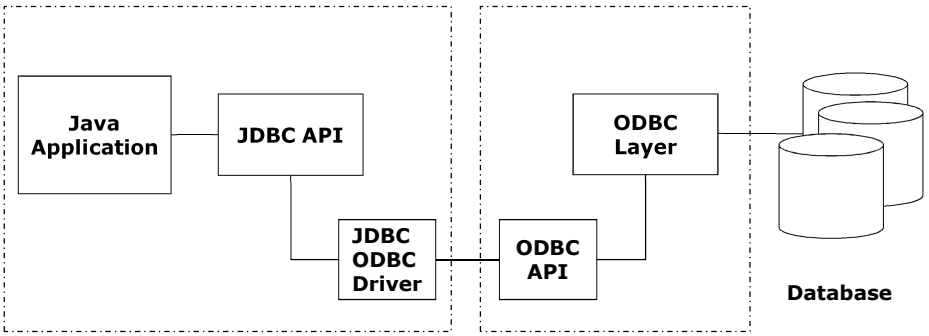
seed
beyond the obvious

JDBC(Drivers)

- JDBC-ODBC Bridge (Type 1)
- Native-API partly Java Driver (Type 2)
- Net-Protocol All-Java Driver (Type 3)
- Native Protocol All-Java Driver (Type 4)

seed
beyond the obvious

JDBC Driver Type 1



Note:- Dash lines indicate process boundaries

seed
beyond the obvious

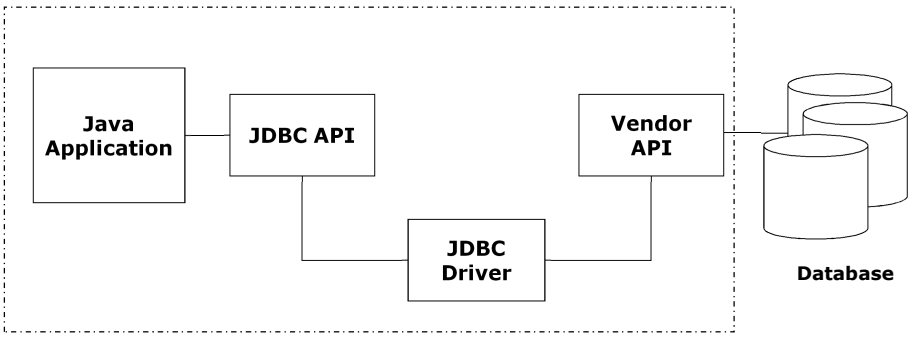
Type 1 Driver (JDBC-ODBC Bridge driver)

- Translates all JDBC API calls to ODBC API calls.
- Relies on an ODBC driver to communicate with the database.

- Disadvantages
  - ODBC required hence all problems regarding ODBC follow.
  - Slow

seed
beyond the obvious

JDBC Driver Type 2



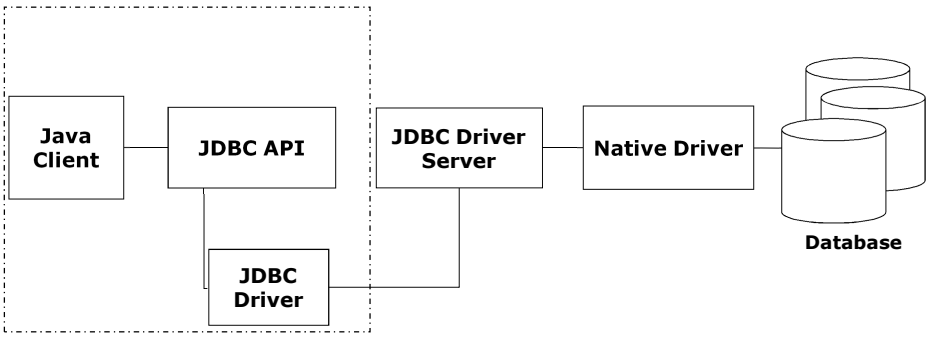
seed
beyond the obvious

Type 2 (Native-API partly Java Driver)

- Written partly in Java & partly in native code, that communicates with the client API of a database.
- Therefore, should install some platform-specific code in addition to Java library.
- The driver uses native 'C' lang lib calls for conversion.

seed
beyond the obvious

JDBC Driver Type 3



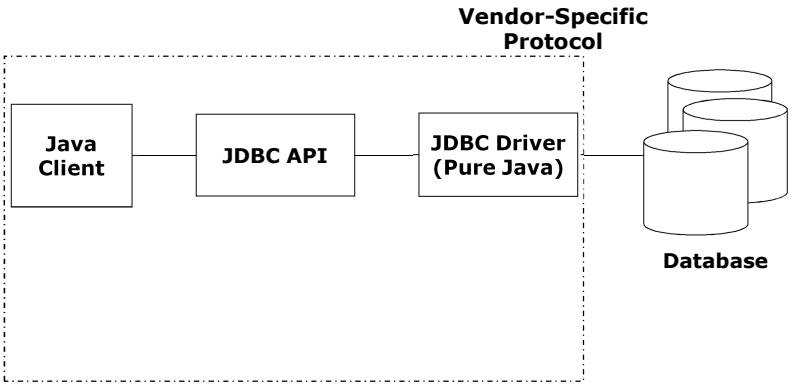
seed
beyond the obvious

Type 3 (Net-Protocol All-Java Driver)

- Uses DB independent protocol to communicate DB-requests to a server component.
- This then translates requests into a DB-specific protocol.
- Since client is independent of the actual DB, deployment is simpler & more flexible.

seed
beyond the obvious

JDBC Driver Type 4



seed
beyond the obvious

Type 4 Driver (Pure Java Drivers)

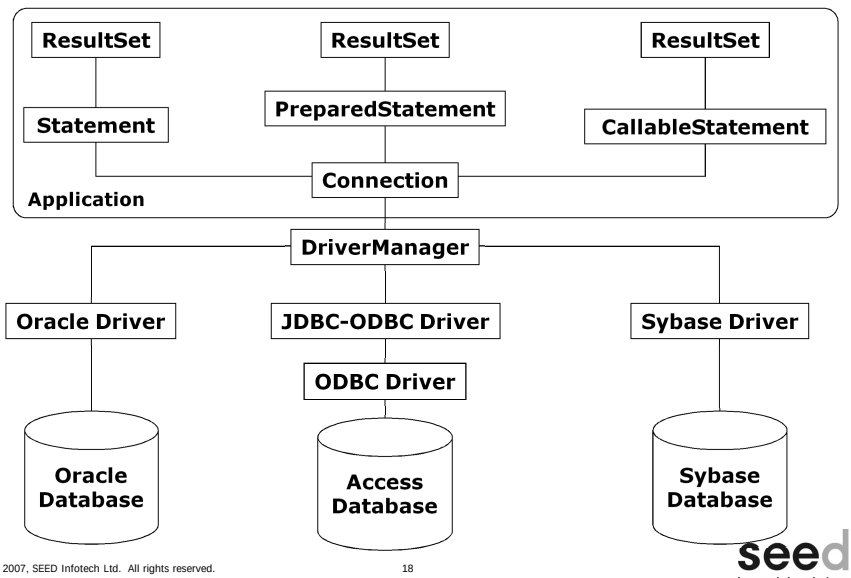
- JDBC calls are directly converted to network protocol used by the DBMS server.
- Driver converts JDBC API calls to direct network calls using vendor-specific networking protocols by making direct socket connections with the DB.
- But driver usually comes only from DB-vendor.

seed
beyond the obvious

JDBC API

- API layer has2 levels of interface.
  - Application layer: developer uses API to make calls to DB via SQL & retrieve results.
  - Driver layer : handles all communication with a specific Driver implementation.

JDBC API



JDBC URL

- Needed by drivers to locate ,access and get other valid information about the databases.
- jdbc:driver:database-name
  - jdbc:Oracle:products
  - jdbc:odbc:mydb; uid = aaa; pwd = secret
  - jdbc:odbc:Sybase
  - jdbc:odbc://whitehouse.gov.5000/cats;

JDBC(Interfaces)

- Driver
- Connection
- Statement
- PreparedStatement
- CallableStatement
- DatabaseMetadata
- ResultSet
- ResultSetMetadata

JDBC(Classes)

- Date
- DriverManager
- DriverPropertyInfo
- Time
- TimeStamp
- Types

seed
beyond the obvious

Driver Interface

- Connection connect(String URL, Properties info)
  - Checks to see if URL is valid.
  - Opens a TCP connection to host & port number specified.
  - Returns an instance of Connection object.
- Boolean acceptsURL(String URL)

seed
beyond the obvious

Driver Manager Class

- Connection getConnection(String URL)
- void registerDriver(Driver driver)
- void deregisterDriver()

- Eg : Connection con = null;
- con =DriverManager.getConnection("jdbc:odbc:mydsn");

seed
beyond the obvious

Connection

- Represents a session with the DB connection provided by driver.
- You use this object to execute queries & , commit or rollback transactions.

seed
beyond the obvious

JDBC(Connection)

- close()
- commit()
- void setAutoCommit(boolean b)
- rollback()
- Statement createStatement()
- CallableStatement prepareCall(String sql)
- PreparedStatement prepareStatement(String sql)
- Demo:- Example1.java

seed
beyond the obvious

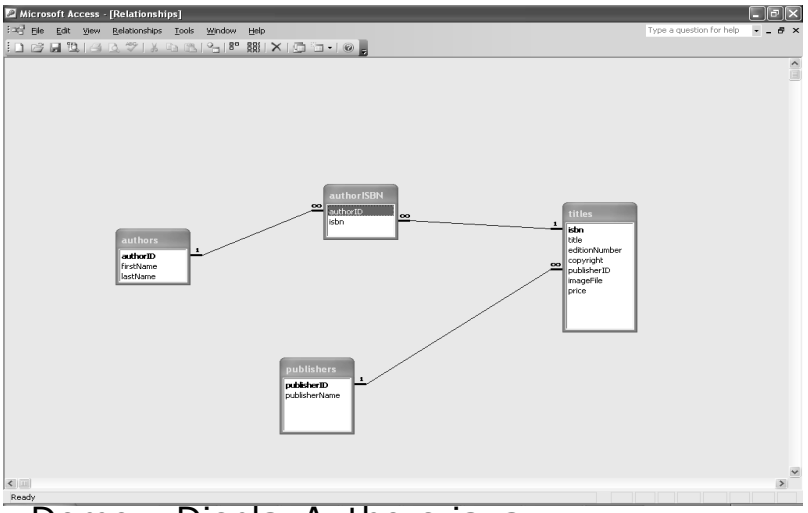
JDBC(Statement)

- Statement
  - PreparedStatement
  - CallableStatement
- Statement Methods
  - boolean execute(String sql)
  - ResultSet executeQuery(String sql)
  - int executeUpdate(String sql)

Demo:- Example2.java

seed
beyond the obvious

JDBC - Demo



Demo:- DisplayAuthors.java

seed
beyond the obvious

JDBC(Parameterised SQL)

String SQL =
"select * from Employees where First_Name=?";

PreparedStatement pstat = con.prepareStatement(sql);

pstat.setString(1, "John");

ResultSet rs = pstat.executeQuery();

pstat.clearParameters();

Demo:- Example4.java

seed
beyond the obvious

JDBC(Stored Procedure)

```
CREATE OR REPLACE PROCEDURE sp_interest
( id  IN  INTEGER
bal  IN  OUT  FLOAT ) AS
BEGIN
   SELECT balance INTO bal FROM      accounts WHERE
   account_id = id ;
   bal := bal + bal * 0.03 ;
   UPDATE accounts
   SET balance = bal
   WHERE account_id = id ;
END;
```


seed
beyond the obvious

JDBC(Stored Procedure)

```
CallableStatement cstmt =
con.prepareCall( "{ call sp_interest(?,?)} ");
cstmt.setInt(1,accountID);
cstmt.setFloat(2, 5888.86);
cstmt.registerOutParameter(2,Types.FLOAT);

cstmt.execute();

System.out.println(cstmt.getFloat(2));
```

Demo:- ProcedureDemo.java

seed
beyond the obvious

ResultSetMetadata Interface

- Object that can be used to find out about the types and properties of the columns in a ResultSet
- Example
  - Number of columns
  - Column title
  - Column type

Demo:- DisplayAuthors.java

seed
beyond the obvious

JDBC(Batch Updates)

```
Statement stmt=con.createStatement();
stmt.addBatch(sql1);
stmt.addBatch(sql2);
stmt.addBatch(sql3);
stmt.executeBatch();
```


seed
beyond the obvious

Quick Recap

- Java application can make use of JDBC API to make a connectivity to database
- There are four types of JDBC drivers
  - Type 1
  - Type 2
  - Type 3
  - Type 4
- JDBC is divided into two layers
  - Application level API
  - Driver level API
- DriverManager is responsible for giving the connection
- To fire an SQL statement we can make use of either a Statement or a PreparedStatement
- CallableStatement is useful for invoking database procedures or functions

seed
beyond the obvious