2019.10.24

24 October 2019 09:02

What is ODBC? ODBC(Open Database Connectivity)开放的数据库连接，由微软开发出来。是一套公共的接口(标准)，用来连接数据库。

❤肃

What is JDBC? JDBC是一套标准(接口)，用来连接【关系型数据库】。各个数据库厂商去实现具体的功能，用户使用时使用相同一套标准即可。但是在使用时得导入不同公司提供的包(里面包含了具体实现)。

JDBC（Java DataBase Connectivity,java数据库连接）是一种用于执行SQL语句的Java API，可以为多种关系数据库提供统一访问，它由一组用Java语言编写的类和接口组成。JDBC提供了一种基准，据此可以构建更高级的工具和接口，使数据库开发人员能够编写数据库应用程序。

JDBC API

JDBC API是什么?

Sun公司提供的一套标准的数据访问接口，用来访问关系型数据库。另外，它包含了一部分类和接口，属于Java语言的一部分。

JDBC可以做什么:

连接数据库

发送sql语句到数据库

处理结果集

JDBC API包含两部分内容:

Java应用程序开发接口(提供给用户)

Java驱动开发接口(实现公共接口功能，各大数据库厂商做)

3.JDBC驱动

其实就是一组类和接口的集合，实现了JDBC相关的功能。

java.sql.Driver 这个驱动接口很重要，我们借助这个类实现了数据库的连接。

JDBC【驱动提供了特定厂商对JDBC API接口类的实现】，驱动必须要提供java.sql包下面这些类的实现：Connection，Statement，PreparedStatement，CallableStatement，ResultSet和Driver.

****

4.JDBC驱动类型

Java中的JDBC驱动可以分为【四种类型】，包括【JDBC-ODBC桥、本地API驱动、网络协议驱动和本地协议驱动】。

5.JDBC-ODBC桥

JDBC-ODBC 桥 是sun公司提供的，是jdk提供的的标准API.【底层是借助ODBC】实现功能，缺陷是效率较低，还得必须安装ODBC驱动。

****以下内容了解即可****

这种类型的驱动实际是把所有 JDBC的调用传递给ODBC ,再由ODBC调用本地数据库驱动代码.(本地数据库驱动代码是指 由数据库厂商提供的数据库操作二进制代码库,例如在oracle for windows中就是oci dll 文 件)

只要本地机装有相关的ODBC驱动那么采用JDBC-ODBC桥几乎可以访问所有的数据库,JDBC- ODBC方法对于客户端已经具备ODBC driver的应用还是可行的.

但是,由于JDBC-ODBC先调用 ODBC再由ODBC去调用本地数据库接口访问数据库.所以,执行效率比较低,对于那些大数据量 存取的应用是不适合的.而且,这种方法要求客户端必须安装ODBC 驱动,所以对于基于 internet ,intranet的应用也是不合适的.因为,你不可能要求所有客户都能找到ODBC driver.

6.本地API驱动

本地API驱动直接【把JDBC调用】转变为【数据库的标准调用】再去访问数据库.效率比第一种高，但得【需要额外的数据库工具】。

这种方法需要本地 数据库驱动代码、本地API驱动、厂商DB代码。这种驱动比起JDBC-ODBC桥执行效率大大提高了.但是,它仍然需要在客户端加载数据库厂商 提供的代码库.这样就不适合基于internet的应用.并且,他的执行效率比起其他的JDBC驱动 还是不够高.

7.网络协议驱动

这种驱动实际上是根据我们熟悉的三层结构建立的.

JDBC先把对数局库的访问请求传递给网络上的【中间件服务器】. 中间服务器再把请求【翻译为符合数据库规范的调用】,再把这种调用 【传给数据库服务器】.

如果中间服务器也是用java开发的,那么在在中间层也可以使用上面说的那俩种驱动类型 JDBC驱动程序作为访问数据库的方法. 网络协议驱动---------中间服务器----------数据库Server

由于这种驱动是基于server的.所以,它不需要在客户端加载数据库厂商提供的代码库.而且 他在执行效率和可升级性方面是比较好的.因为大部分功能实现都在server端,所以这种驱动 可以设计的很小,可以非常快速的加载到内存中.

但是,这种驱动在中间件层仍然需要有配置 其它数据库驱动程序,并且由于多了一个中间层传递数据,它的执行效率还不是最好.

8.本地协议驱动

我们【当前使用的JDBC驱动】一般都是这种类型。

这种类型的驱动【完全由java实现】,实现了平台独立性。

它可以直接把JDBC调用转换为符合相关数据库系统规范的请求.这种驱动写的应用可以直接和数据库服务器通讯。

本地协议驱动【效率非常高】,因为它不需要先把JDBC的调用传给ODBC或本地数据库接口或者是中间层服务器。

而且,它不需要在客户端或服务器端装载任何的软件或驱动， 这种驱动程序可以动态的被下载。

但是对于不同的数据库【需要下载不同的驱动程序】(驱动实现jar包)。

****

9.JDBC数据库操作相关类和接口

Driver驱动接口

DriverManager驱动管理器【类】

Connection数据库连接

Statement操作语句

PreparedStatement
CallableStatement

ResultSet结果集

DatabaseMetadata数据库信息(包含数据库有哪些表，视图，列，类型等信息，由表反向生成类)

ResultSetMetadata结果集信息

Types类型【类】(数据库表类型与java代码中类型String,int,double等的对应)

jdbc开发接口，存在于两个包下，其中【java.sql包】下实现了基本功能；剩余部分存在于【javax.sql包】中，实现扩展功能。

10.JDBC操作流程

用户如何通过Java代码连接到 指定的数据库服务器，然后进行操作。

有6个固定的步骤，按照步骤操作即可。

1).注册驱动(驱动：具体功能实现)

2).建立数据库连接

3).创建Statement对象(可以进行sql与执行)

4).执行SQL语句(增删改查)

5).处理结果集

6).关闭相应资源

11.注册驱动

【注册驱动】的过程，可以理解成【安装驱动】。想一下生活中安装好系统，usb外接鼠标不能用，安装个驱动就可以用了。此处注册好了驱动，数据库连接就可以使用了。

注册驱动的三种方案:

使用类加载器

实例化一个驱动对象，使用new的方式

使用property

ojdbc5.jar Oracle11g配套推出的

ojdbc6.jar 11g配套推出

ojdbc14.jar Oracle10g发布时配套推出的

14、5、6与JDK的版本有关系，14是1.4版本以上，ojdbc5.jar得jdk5,ojdbc6.jar得用jdk6及其以上版本。

查看自己jdk版本命令: java -version

此后学习过程中，不仅仅要注意代码，还要关注环境是否配置正常。(防火墙、杀毒软件等都可能会影响最后的结果)

12.JDBC连接类型

JDBC的连接类型有两种，分别是oci 和 thin

1)、JDBC OCI

oci是oracle call interface的缩写，此驱动类似于传统的ODBC 驱动。

因为它需要Oracle Call Interface and Net8，所以它【需要】在运行使用此驱动的JAVA程序的机器上安装【客户端软件】，其实主要是用到orcale客户端里以dll方式提供的oci和服务器配置。

2)、JDBC Thin

thin是for thin client的意思，这种驱动一般用在运行在WEB浏览器中的JAVA程序。

它不是通过OCI or Net8，而是通过Java sockets进行通信，是纯java实现的驱动，因此【不需要】在使用JDBC Thin的客户端机器上安装orcale【客户端软件】，所以有很好的移植性，通常用在web开发中。

我们更多的【选择thin】这种方式进行服务器连接。

13.JDBC四要素

1).String driver = "jar包Driver类全包名";

告诉我们具体数据库驱动在哪里，方便后期注册。

String dirver = "oracle.jdbc.driver.OracleDriver";

2).URL 确定连接到那个具体的数据库

jdbc:oracle:oci:@<SID> oci连接方式

jdbc:oracle:thin:@<SID> thin连接方式

String url = "jdbc:oracle:thin:@127.0.0.1:1521:XE";

3).String user = "用户名";

4).String password = "密码";

14.注册driver和连接数据库

14.1 注册driver有三种方式

a.使用类加载器

```
String driverName = "oracle.jdbc.driver.OracleDriver";
Class.forName(driverName);
```

b.实例化一个驱动对象

实例化驱动对象

```
Driver driver = new oracle.jdbc.driver.OracleDriver();
```

注册驱动

```
DriverManager.registerDriver(driver);
```

c.使用property

```
System.setProperty("jdbc.drivers", "oracle.jdbc.driver.OracleDriver");
```

使用【系统配置】进行驱动注册，其中【第一个参数值固定】，第二个参数为驱动类的全包名。

补充: 也可以运行虚拟机时，设置运行参数。

这种方式也是通过系统配置进行驱动注册。

```
-Djdbc.drivers=oracle.jdbc.driver.OracleDriver
```

14.2 数据库连接的方式有两种:

第一种是利用DriverManager.getConnection(url, user, passwd);

第二种是利用Driver类直接连接(其实第一种底层也是通过Driver建立的连接)。

```
Driver driver = new oracle.jdbc.driver.OracDriver();
Properties info = new Properties();
info.setProperty("user","");
info.setProperty("password","");

conn = driver.connect(url,info);
```

关于DriverManager和Driver的关系

一个DriverManager可以管理多个Driver，在具体进行数据库连接时，管理类借助Driver里面的connect方法建立连接。

****

数据库连接不成功解决方案

反复检查四要素是否写错

数据库未启动

防火墙影响

换个jar包试试

如果安装oracle出现中文路径，不行

如果安装成功后修改计算机用户名，不行

重装oracle
****

15.Statement语句
Statement stmt = conn.createStatement();
String sql = "";

stmt.execute(sql);
stmt.executeQuery(sql);
stmt.executeUpdate(sql);

一共有三种执行方式，他们都可以正常执行select insert update delete语句，区别在于返回值类型。

execute(sql)返回boolean，执行【返回结果集则true】,否则false;
executeUpdate(sql)【返回int】,执行影响了多少条数据;
executeQuery(sql)返回结果集ResultSet。

16.java.util.Date和java.sql.Date
java.util.Date是java.sql.Date的父类;
一般情况下表示时间，用util.Date,在操作sql语句时才使用sql.Date
通过【时间戳】值进行转换
例如： new java.sql.Date(utilDate.getTime());

17.事务提交
DML语句

设置事务自动提交 关闭，【默认是开启】的
conn.setAutoCommit(false);
...创建stmt对象
...执行sql语句
conn.commit();手动提交事务
conn.rollback();回滚事务

注意:【conn.close();事务会自动提交】。

```
create table student(
        id number primary key,
        name varchar2(20) not null,
        birthday date,
        score number
);
```

同构：sql语句结构相同
插入100条数据

```
insert into student(id,name,birthday,score)
values(?,?,?,?);
```

PrepareStatement处理同构的sql语句

异构: 结构不同
insert into ;
delete;
update;

Statement来处理异构的sql语句。

//同构 ps执行原理
//1.构建ps对象的同时，先将sql语句发送到DB服务器
//2.将具体的值设置给ps
//3.ps执行，传递数据给DB服务器
//4.DB服务器接收数据，结合sql框架，完成功能
//5.返回数据给当前应用程序

//1.将sql语句发送到 DB数据库，
//2.DB数据库执行sql语句
//3.将结果返回给当前应用程序

18.同构和异构

【异构】sql语句，结构不同的语句。

Statement语句执行，每次执行sql语句，都会把sql语句通过网络发送到服务器，然后让服务器实现功能。

同构 【结构相同】的sql语句

```
insert into student(id,name,age,birthday)
values(x,y,z,m);

insert into student(id,name,age,birthday)
values(x,y,z,m);
```

在实际使用时，可以提前将Sql语句发送给数据库进行【预编译】，然后每次只需要传输值即可，这样子可以提高效率。

```
String sql = "insert into student(id,name,age,birthday)
values(?,?,?,?)";
ps = conn.prepareStatement(sql);

ps.setInt(1,7);
ps.setString(2, "王五");
ps.setInt(3, 21);
```

注意，【将 util.Date转换为sql.Date】 通过时间戳值进行

```
java.util.Date utilDate = new java.util.Date();
ps.setDate(4, new java.sql.Date(utilDate.getTime()));
```

注意，ps执行sql语句时，没有必要也不能传sql语句，因为sql语句已经提前发送到服务器了

ps.execute();

create table student(
id number primary key,
name varchar2(20) not null,
birthday date,
score number
);

19.通过PrepareStatement进行其他操作

PrepareStatement可以预编译sql语句。

使用PrepareStatement语句进行增删改查。

使用PrepareStatement的好处，除了效率高，还有就是 【省去了字符串的拼接】。

在select时，将数据导入到Student对象中，然后添加到list集合中，最后统一遍历。

****

注意：如果同时在终端和代码里面操作sql语句，操作后一定要提交事务，否则会造成线程阻塞，代码一直运行不成功。

驱动注册的 4种方式

1.Class.forName();
2.OracleDriver od = new OracleDriver();
DriverManager.registerDriver(od);
3.系统配置
System.setProperty("jdbc.drivers","");
4.配置虚拟机运行时参数
-Djdbc.drivers=...;

数据连接的 2种方式

1.DriverManager.getConnection(url,user,passwd);
2.od.connect(url,info);
inof = new Properties();

事务的操作

终端 delete from student;(开启事务)
commit;
源码 insert into ;(开启事务)
线程阻塞，由事务未提交引起的。

java.util.Date和java.sql.Date

sql下的Date 是 util的子类
时间戳 long值 毫秒

PrepareStatement和Statement

同构 异构