**JDBC技术**

# 概述

**JDBC: Java数据库连接（Java Database Connectivity）**

## 概念

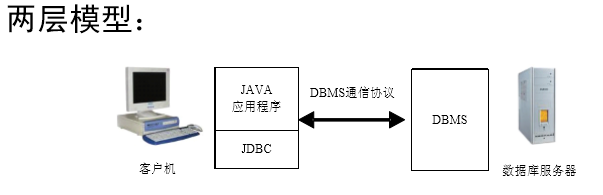
JDBC是一种用于执行SQL语句的Java API，它由一组用Java编程语言编写的类和接口组成

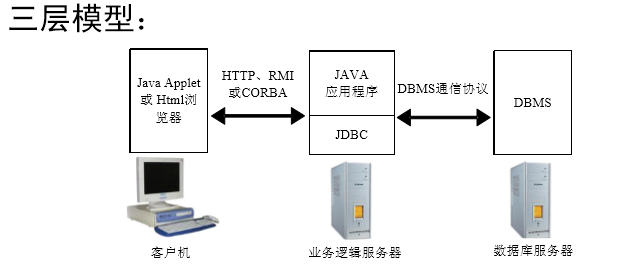
## 特性

JDBC具有简单、健壮、安全、可移植、获取方便等特点

## 支持

JDBC既支持数据库访问的两层模型，也支持三层模型





## 驱动类型

① JDBC-ODBC桥加ODBC驱动程序

② 本地API部分用Java来编写的驱动程序

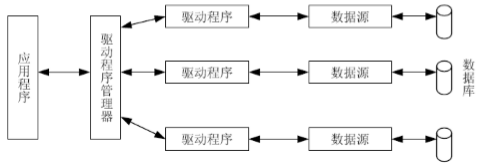
③ JDBC网络纯Java驱动程序

④ 本地协议纯Java驱动程序

## ODBC

**ODBC (**Open Database Connectivity **开放式数据库连接**)**：**ODBC API是使用最广的、用于访问关系数据库的编程接口，它可以将所有平台的关系型数据库连接起来

## 纯ODBC模型结构



## JDBC-ODBC模型结构

# 连接数据库

## 使用JDBC访问数据库的基本步骤

a.加载JDBC驱动程序

b.建立数据库连接

c.创建Statement对象

d.执行SQL语句

f.处理返回结果

g.关闭创建的对象

## 加载JDBC驱动程序

**1）加载驱动程序的方法：**

**Class.forName(＂DriverName＂);**

DriverName是要加载的JDBC驱动程序名称

**2）加载JDBC-ODBC数据库驱动程序的方法：**

**Class.forName(＂sun.jdbc.odbc.JdbcOdbcDriver＂);**

**3)加载Oracle数据库驱动程序的方法：**

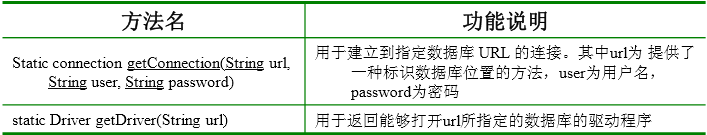
**Class.forName(＂oracle.jdbc.driver.OracleDriver＂);**

## 创建数据库连接（使用DriverManager）

**1）概念：**

DriverManager类是JDBC的管理层，作用于用户和驱动程序之间，跟踪可用的驱动程序，并在数据库和相应驱动程序之间建立连接。

**2）常用方法：**



**3）使用DriverManager类的getConnection()创建一个数据库连接**

**Connection conn = DriverManager.getConnection(String url,String user,String password);**

## 连接Oracle的URL

**jdbc:oracle:thin:@serverName:port:instance**

oracle为子协议名称

thin是oracle数据库的一种连接方式

serverName为oracle数据库服务器名称，可以是一个域名，也可以是IP地址

port为oracle数据的端口号，默认是1521

instance是数据库的实例名

示例：

**Class.forName(＂oracle.jdbc.driver.OracleDriver＂);**

**Connection Conn = DriverManager.getConnection(**

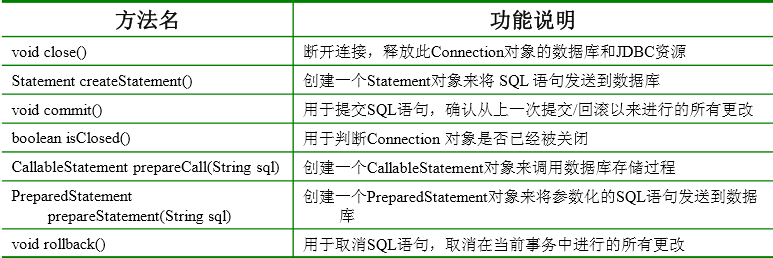
**＂jdbc:oracle:thin:@localhost:1521:orcl＂,＂scott＂,＂tiger＂);**

## Connection

**1)Connection是一个接口，表示与数据库的连接，并拥有创建SQL语句的方法，以完成基本的SQL操作，同时为数据库事务处理提供提交和回滚的方法**

一个应用程序可与单个数据库建立一个或多个连接，也可以与多个数据库建立连接

**2)Connection接口的常用方法**



## Statement

**1) 创建完连接之后，可以通过此连接向目标数据库发送SQL语句。**

**在发送SQL语句之前，需创建一个Statement类的对象，该对象负责将SQL语句发送给数据库。如果SQL语句运行后返回产生结果集，Statement对象会将结果集封装成ResultSet对象并返回**

**2)通过调用Connection接口的createStatement方法来创建Statement对象**

**Statement smt = conn.createStatement();**

**3)执行SQL语句，获取Statement对象之后，就可以使用该对象的executeQuery()方法来执行SQL语句：**

**ResultSet rs = smt.executeQuery(＂select \* from student＂);**

**4)executeQuery()方法返回一个ResultSet对象，它包含了SQL的查询结果**

## 处理返回结果

**1)在JDBC中，SQL的查询结果使用ResultSet封装**

**2)ResultSet对象维持了执行某个SQL语句后满足条件的所有的行，并提供了一系列访问结果集中数据的方法**

**3)通过调用ResultSet对象的方法，处理返回结果**

## 关闭创建的对象

**1)当数据库操作执行完毕或退出应用前，需将数据库访问过程中建立的对象按顺序关闭，防止系统资源浪费**

**2)关闭的顺序：**

* + - * **关闭结果集**
      * **关闭Statement对象**
      * **关闭连接**

# 操作数据库

**在JDBC中要执行SQL查询语句可以通过执行一般查询、参数查询和存储过程三种**

## Statement

**a)用于执行静态的SQL语句，静态的SQL语句不需要接收任何参数，Statement主要功能是将SQL语句传送给数据库，并将SQL语句的执行结果返回**

**b)提交的SQL语句可以是：**

**select查询语句**

**SQL DML语句：INSERT、UPDATE、DELETE**

**SQL DDL语句：CREATE　TABLE、DROP TABLE**

**c)Statement接口的常用方法：**



**d)Statement接口提供了三种执行SQL语句的方法：**

**executeQuery()：用于产生单个结果集的语句，如：SELECT**

**executeUpdate()：用于执行INSERT、UPDATE、DELETE语句以及SQL DDL语句，如CREATE TABLE、DROP TABLE**

返回值是一个整数，指示受影响的行数

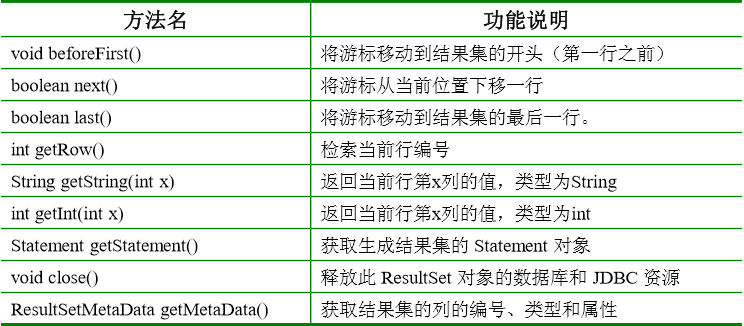
对于CREATE TABLE、DROP TABLE等语句，返回值总为零

**execute()：用于执行返回多个结果集、多个更新计数或二者的组合**

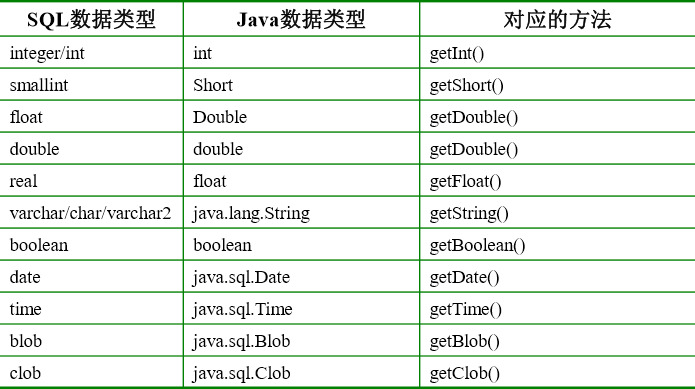
方法比较特殊，一般是在用户不知道执行SQL语句后会产生什么结果或可能有多种类型的结果产生时才会使用

**f)ResultSet封装了Statement的executeQuery()方法返回的结果集**

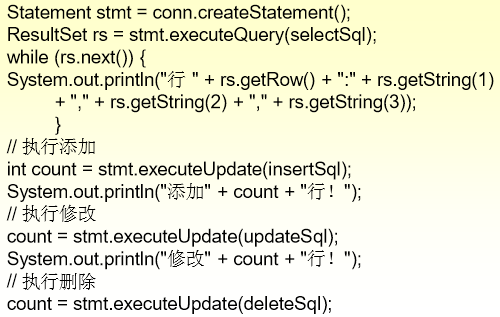
**g)ResultSet常用方法：**



**h)ResultSet提供了一套getXXX()方法，取值时要注意数据库的字段数据类型和Java数据类型之间的匹配（可通过列名或列号标识要获取数据的列）：**



**示例：StatementDemo.Java**



## PreparedStatement

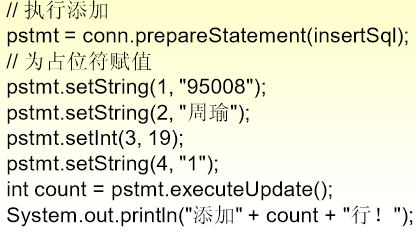
**a)PrepareStatement接口是Statement接口的子接口，它继承了Statement的所有功能**

**b)特点：执行速度要快于Statement对象；SQL语句中提供参数，提高程序的灵活性和执行效率**

**c)语法格式：PrepareStatement对象通过Connection的prepareStatement()方法创建；SQL语句中的参数使用“？”作为站位符**



**d)在执行带参数的SQL语句前，必须对“？”赋值；通过setXXX()方法**



## CallableStatement

**a)存储过程是数据库中已经存在的SQL语句。在JDBC中执行数据库中的存储过程可以使用CallableStatement对象**

**b)CallableStatement继承了Statement和PrepareStatement,**

**c)特点：可以处理一般的SQL语句，也可以处理出入参数，同时还定义了OUT（输出）参数以及INOUT（输入输出）参数的处理方法**

**d)CallableStatement对象可以使用Connection类的prepareCall()方法**

**f)存储过程有参数，可以使用占位符“？”代替：**



**g)在执行一个存储过程之前，必须先对其中的OUT参数进行类型注册**

**registerOutParamenter(int index,int sqlType);**

第一个参数为对应存储过程占位符

第二个参数为对应占位符的变量类型，通过java.sql.Type的静态常量来指定：Types.FLOAT、Types.INTEGER等

**h)当使用getXXX()方法获取OUT参数的值时，XXX对应的Java类型必须与所注册的SQL类型相符**

**i)存储过程有返回值时调用方式：**



# 元数据（Meta Data）

**a)有关数据库和表结构的信息**（数据库中的表、表的列、表的索引、数据类型、对SQL的支持程度等信息）

**b)元数据类型：DatabaseMetaData；ResultSetMetaData**

## DatabaseMetaData

**a)DatabaseMetadata接口主要是用来得到关于数据库的信息**

**b)DatabaseMetaData对象是通过Connection的getMetaData()方法获取**

**c)分类：返回值类型为boolean型，用于检查数据库或驱动器是否支持某项功能**

**用于获取数据库或驱动器本身的某些特征值**



# ResultSetMetaData

**a)ResultSetMetaData接口主要用来获取结果集的结构**

**b)ResultSetMetaData对象通过ResultSet的getMetaData()方法获取**

**c)ResultSetMetaData的常用方法和功能：**



# 事务操作

**事务是构成单一逻辑工作单位的操作集合。已提交事务是指成功执行完毕的事务，未能成功完成的事务称为中止事务，对中止事务造成的变更需要进行撤销处理，称为事务回滚**

**a)事务操作缺省是自动提交：**

一条对数据库的更新操作成功后，系统将自动调用commit()方法提交

失败调用rollback()方法来回滚。

**b)为了能将多个JDBC语句组合成一个操作单元，以保证数据的一致性。可以禁止自动提交，之后把多个数据库操作的表达式作为一个事务，在操作完成后调用commit()来进行整体提交，倘若其中一个表达式操作失败，可以在异常捕获时调用rollback()进行回滚。**

**c)利用Connection对象的setAutoCommit()方法，可以开启或关闭自动提交方式。**

# 小结

* **JDBC是Java应用于数据库通信的基础**
* **JDBC包含一组类与接口，用于连接到任何数据库**
* **JDBC访问数据库的一般步骤是：**
  + - 加载JDBC驱动程序
    - 建立数据库连接
    - 创建Statement对象
    - 执行SQL语句
    - 处理返回结果
    - 关闭创建的对象
* **JDBC通过ResultSet维持查询结果集，并提供游标进行数据操作**
* **通过Statement实现静态SQL查询（不带参数）**
* **使用PrepareStatement实现动态SQL查询（带参数）**
* **使用CallableStatement实现存储过程的调用**
* **DatabaseMetaData接口用于得到相关数据库的信息**
* **ResultSetMetaData接口主要用来获取结果集的结构**
* **JDBC默认的事务提交模式是自动提交**
* **通过setAutoCommit()方法控制自动提交模式，使用rollback()方法实现事务回滚**