第 11 章　JDBC 数据库

11.1　JDBC 简介

在 Widow　和 Macntosh 平台上各种数据库的通信中，开放数据连接（ODBC）作为一种事实上的标准而出现。Java 数据库连接 JDBC 与 ODBC 的概念和标准一样，两者都是为各种流行数据库提供无缝连接技术的。作为各种数据库间的通信手段，ODBC 已被广大开发商和编程人员认为是当今各种数据库间进行通信的较好的方法。这些数据库包括：Sybas SQL Server、Mirosoft SQL Server 、Mirosoft A　　、Paradox、Orac 、xBas 等等。

ODBC 是图形用户界面前端和数据库后台之间的通信层，它使得数据源所指定的函数或命令是透明的，使用标准的 SQL 语句便可以访问数据库而不需要使用特定的数据库命令。ODBC 使用户程序具有可移植性，即不需要作任何修改或只作很小的修改就可以将应用程序中的数据库从一种移植到另一种，例如从 Mirosoft A　　到 Mirosoft SQL Server 。JDBC 同 ODBC 完全类似，它利用 java.sql 包及其接口、类和异常事件，通过 JDBC-ODBC 桥可以对数据库进行操作，但数据库本身没有什么要求，不需要什么特别的命令，只需要同 ODBC 一样在服务器中设置 ODBC 数据源。

11.2　数据库类实现

JDBC 是由 Java.sql 类包实现的，这个包中含有所有的 JDBC 类和方法，如表 11-1 所示。

表 11-1　　JDBC 类

类型	类
驱动程序	java.qlDrive java.qlDriveManager java.qlDrivePropertyInfo
连接	java.qlti
语句	java.qlStatet java.qlPreparedStatet java.qlalablStatet
结果集	java.qlReultSet
错误/警告	java.qlSQLExcpti java.qlSQLWarng
元数据	java.qlDatabasMetaData java.qlReultSetMetaData
日期/时间	java.qlDate java.qlTi java.qlTiStamp
其他	java.qlType java.qlDataTrunati

java.sql.Driver 类通常用来得到驱动程序的 PropertyInfo、版本号等信息，在一个使用 JDBC 的 Java 程序的执行过程中，该类可以被调用多次。下表列出了该类的主要方法：

publiynchronized Connetion connt(String url,Properti properti)
throws SQLException
建立同数据库的连接；
publi boolean acptsURL(String url) throws SQLException
如果驱动程序认为可以打开指定数据库 URL 的连接则返回 true；
publiynchronized DriverPropertyInfo[] getPropertyInfo(String url,Properti properti
返回驱动器属性信息；
publint getMajorVerson()
返回驱动程序的主版本号；
publint getMinorVerson()
返回驱动程序的次版本号。
publi boolean jdbcCpliant()

DriverManager 保持驱动程序的信息、状态信息等，当每个驱动程序被装入时，它用 DriverManager 进行登记，当需要打开一个连接时，DriverManager 根据 JDBCPI 选择所需要的驱动程序，下面列出该类的主要方法。

publitatiynchronized Connetion getConntion
(String url, Propertinfo) throws SQLException
publitatiynchronized Connetion getConntion
(String url,String user, String pasord) throws SQLException
建立与指定数据库的连接。
publitatiynchronized Connetion getConntion(String url) throws SQLException
视图建立和指定数据库的连接。
publitati Driver getDriver(String url) throws SQLException
视图定位一个指定 url 的驱动程序。
publitatiynchronized void registerDriver(Driver driver) throws SQLException
注册指定的驱动程序。
publitati void deregisterDriver(Driver driver) throws SQLException
取消注册指定的驱动程序。
publitati Enumration getDrivers()
publitati void stLoginTiout(int sonds)
设置所有驱动程序可以等待试图连接数据库的最大时间。
publitatint getLoginTiout()
得到所有驱动程序可以等待试图连接数据库的最大时间。
publitati void stLogStream(PrintStream out)
设置被所有驱动程序所有的跟踪打印流；

publitati PrintStream getLogStream()
publitati void println(String mage)
打印指定信息到当前 JDBCog　流。

11.3　访问数据库的 JDBC 类

11.3.1　Connetion　类

Conntion　类是 JDBC 类中的主要类之一，也是在 Java 中访问数据库的必不可少的类之一，它包含了从事务处理到创建语句等许多功能。Ction　　类中的方法如下所示：

publiynchronized Statent createStatent() throws SQLException
申明没有参数的 SQL 对象；
publiynchronized PreparedStatent prepareStatent(String sql) throws SQLException
创建 sql 语句对象；
publiynchronized CalableStatent prepareal(String sql) throws SQLException
创建 sql 语句对象；
publi String nativeSQL(String sql) throws SQLException
publiynchronized void stAutoComt(booleannableutoComt) throws SQLException
设置自动提交方式；
publi boolean getAutoComt() throws SQLException
得到当前自动提交状态；
publiynchronized void ct() throws SQLException
提交上次永久提交或回滚后所发生的变化并释放该 connetion　占据的被锁数据库；
publiynchronized void rollback() throws SQLException
回滚事务；
publiynchronized void cose() throws SQLException
强制关闭 JDBC 资源；
publi booleanosed() throws SQLException
返回该 conntion　是否被关闭；
publiynchronized DatabasMetaData getMetaData() throws SQLException
返回该 conntion　的 DatabasMetaData　对象；
publiynchronized void stReadOnly(boolean readOnly) throws SQLException
设置 connetion　的读取方式；
publi booleanReadOnly() throws SQLException
返回该 conntion　是否是只读；
publiynchronized void stCatalog(String catalog) throws SQLException
设置目录；

publi String getCatalog() throws SQLException

返回当前 cnnetion 的目录名；

publiynchronized void stTransactionIsolation(int lve) throws SQLExcepti on

改变事务隔离水平；

publint getTransactionIsolation() throws SQLException

当前 connetion 的事务隔离方式；

publi SQLWarning getWarnings() throws SQLException

该 connetion 返回后得到的第一个 warning；

publiynchronized void carWarnings() throws SQLException

调用该方法以后，在调用 getWarnings()方法将返回一个 nul 值直到该 ction 出现一个新的 warning。

设在源机器上已经配置好 ODBC 数据源“ tep ”，则利用下面的代码可以建立与该数据源的 Conntion 连接：

```
Cas.forNam("sun.jdbc.odbcJdbcOdbcDriver");
Conntion connetion= DriverManager.getConnetion("jdbc:odbc:tep");
```

其中 ODBC 数据库的 JDBC URL 格式为：

```
jdbc:odbc:<ODBC DSN>;Usr=<usernam>;pwd=<pasord>
```

Conntion 对象执行的另一个重要功能就是事务管理，事务的处理依赖于一个内部自动递交标志的状态，这个标志可以通过 cnnetion 对象的 stAutoComt() 方法来进行设定，可以通过 getAutot() 方法读出。当这个标志为 true 时，事务一完成就会自动递交，不需要 Java 应用程序进行任何干涉；当这个表示为 fal 时，系统对事物的处理处于手动方式下，java 应用程序有可选项使用 ct() 和 rollback()方法递交上次递交之后发生的事务的集合或者回滚这些事务。

11.3.2 DatabasMetaData 类

DatabasMetaData 类似于 ODBC 中的目录函数，应用程序用它来查询底层 DBMS 系统的表格并且获取信息。DatabasMetaData 对象和它的方法可以给出很多底层数据库的信息，这些信息对数据库工具、自动数据转换和网关程序十分重要。它的主要方法如下所示：

```
publiynchronized boolean alProcedurerealable() throws SQLException
publiynchronized boolean alTablereSetable() throws SQLException
```

返回所有的表是否被当前的用户选中；

```
publi String getURL() throws SQLException
```

返回数据库的 URL；

```
publi String getUsrNam() throws SQLException
publi booleanReadOnly() throws SQLException
```

publi boolean nulreSortedHigh() throws SQLException
publi boolean nulreSortedLow() throws SQLException
publi boolean nulreSortedAtStart() throws SQLException
publi boolean nulreSortedAtEnd() throws SQLException
publi String getDatabasProductNam() throws SQLException
返回数据库产品名；
publi String getDatabasProductVerson() throws SQLException
返回数据库产品版本；
publi String getDriverNam() throws SQLException
返回 JDBC 驱动程序名；
publi String getDriverVerson() throws SQLException
返回 JDBC 驱动程序版本；
publint getDriverMajorVerson()
返回驱动程序的主版本号；
publint getDriverMinorVerson()
返回驱动程序的次版本号；
publi boolean useLocalFi() throws SQLException
数据库存储表是否在本地文件中；
publi boolean useLocalFiPerTable() throws SQLException
publi booleanupportsMixedCasIdentifirs() throws SQLException
返回是否支持混合字母标识
publi booleantoreUpperCasIdentifirs() throws SQLException
publi booleantoreLowerCasIdentifirs() throws SQLException
publi booleantoreMixedCasIdentifirs() throws SQLException
返回 store 是否对字母的大小写或混合表示区分
publi booleanupportsMixedCasQuotedIdentifirs() throws SQLException
publi booleantoreUpperCasQuotedIdentifirs() throws SQLException
publi booleantoreLowerCasQuotedIdentifirs() throws SQLException
publi booleantoreMixedCasQuotedIdentifirs() throws SQLException
publi String getIdentifirQuoteString() throws SQLException
publi String getSQLKeywords() throws SQLException
publi String getNumeriFunctions() throws SQLException
publi String getStringFunctions() throws SQLException
publi String getSysteFunctions() throws SQLException
publi String getTiDateFunctions() throws SQLException
publi String getSearchStringEsape() throws SQLException
publi String getExtraNamharacters() throws SQLException
publi booleanupportsterTableWithAddColumn() throws SQLException
publi booleanupportsterTableWithDropColumn() throws SQLExcepti
publi booleanupportsolumnAlasng() throws SQLException

```
publi boolean nulPlusNonNullIsNull() throws SQLException
publi booleanupportsonvert() throws SQLException
publi booleanupportsonvert(int froType,int toType) throws SQLException
publi booleanupportsTableorreationNam() throws SQLException
publi booleanupportsDifferentTableorreationNam() throws SQLException
publi booleanupportsExpreonsInOrderBy() throws SQLException
publi booleanupportsOrderByUnreated() throws SQLException
publi booleanupportsGroupBy() throws SQLException
publi booleanupportsGroupByUnreated() throws SQLExcepti
publi booleanupportsGroupByBeyondSet() throws SQLException
publi booleanupportsLikeEsapeCause() throws SQLException
publi booleanupportsMultipleReultSets() throws SQLException
publi booleanupportsMultipleTransactions() throws SQLException
publi booleanupportsNonNullableolumns() throws SQLException
publi booleanupportsMinimumSQLGramar() throws SQLException
publi booleanupportsoreSQLGramar() throws SQLException
publi booleanupportsExtendedSQLGramar() throws SQLExcept      ion
publi booleanupportsNSI92EntryLeveSQL() throws SQLException
publi booleanupportsNSI92IntermdiateSQL() throws SQLException
publi booleanupportsNSI92FulSQL() throws SQLException
publi booleanupportsIntegrityEnhancentFacty() throws SQLException
publi booleanupportsOuterJoins() throws SQLException
publi booleanupportsFullOuterJoins() throws SQLException
publi booleanupportsLitedOuterJoins() throws SQLException
publi String getSchemaTerm() throws SQLException
publi String getProcedureTerm() throws SQLException
publi String getCatalogTerm() throws SQLException
publi booleanatalogAtStart() throws SQLException
publi String getCatalogSeparator() throws SQLException
publi booleanupportsSchemasInDataManipulation() throws SQLException
publi booleanupportsSchemasInProcedureal() throws SQLException
publi booleanupportsSchemasInTableDefinitions() throws SQLException
publi booleanupportsSchemasInIndexDefinitions() throws SQLException
publi booleanupportsSchemasInPrivigeDefinitions() throws SQLException
publi booleanupportsatalogsInDataManipulation() throws SQLException
publi booleanupportsatalogsInProcedureal() throws SQLException
publi booleanupportsatalogsInTableDefinitions() throws SQLException
publi booleanupportsatalogsInIndexDefinitions() throws SQLException
publi booleanupportsatalogsInPrivigeDefinitions() throws SQLException
publi booleanupportsPositionedDete() throws SQLException
```

```
publi booleanupportsPositionedUpdate() throws SQLException
publi booleanupportsSetForUpdate() throws SQLException
publi booleanupportsStoredProdure() throws SQLException
publi booleanupportsSubqueriInComparions() throws SQLException
publi booleanupportsSubqueriInExists() throws SQLException
publi booleanupportsSubqueriInIns() throws SQLException
publi booleanupportsSubqueriInQuantifids() throws SQLException
publi booleanupportsorreatedSubqueri() throws SQLException
publi booleanupportsUnion() throws SQLException
publi booleanupportsUnionA() throws SQLException
publi booleanupportsOpenursorsrosst() throws SQLException
publi booleanupportsOpenursorsrossRollback() throws SQLException
publi booleanupportsOpenStatentsrosst() throws SQLException
publi booleanupportsOpenStatentsrossRollback() throws SQLException
publint getMaxBinaryLiteralLength() throws SQLException
publint getMaxCharLiteralLength() throws SQLException
publint getMaxColumnNamLength() throws SQLException
publint getMaxColumnsInGroupBy() throws SQLException
publint getMaxColumnsInIndex() throws SQLException
publint getMaxColumnsInOrderBy() throws SQLException
publint getMaxColumnsInSet() throws SQLException
publint getMaxColumnsInTable() throws SQLException
publint getMaxConnetions() throws SQLException
publint getMaxCursorNamLength() throws SQLException
publint getMaxIndexLength() throws SQLException
publint getMaxScaNamLength() throws SQLException
publint getMaxProcedureNamLength() throws SQLException
publint getMaxCatalogNamLength() throws SQLException
publint getMaxRowSize() throws SQLException
publi boolean doeMaxRowSizeIncludeBlobs() throws SQLException
publint getMaxStatentLength() throws SQLException
publint getMaxStatents() throws SQLException
publint getMaxTableNamLength() throws SQLException
publint getMaxTableInSet() throws SQLException
publint getMaxUsrNamLength() throws SQLException
publint getDefaultTransactionIsolation() throws SQLException
publi booleanupportsTransactions() throws SQLException
publi booleanupportsTransactionIsolationLeve(int lve) throws SQLException
publi booleanupportsDataDefinitionAndDataManipulationTransactions()
        throws SQLException
```

```
publi booleanupportsDataManipulationTransactionsOnly() throws SQLException
publi boolean dataDefinitionCauseTransactionCt() throws SQLException
publi boolean dataDefinitionIgnoredInTransactions() throws SQLException
publiynchronized ReultSet getProcedure(String catalog,
        String shemaPattern ,String procedureNamPattern) throws SQLException
publiynchronized ReultSet getProcedureolumns(String catalog,
        String shemaPattern, String procedureNamPattern,
        String columnNamPattern) throws SQLException
publiynchronized ReultSet getTable(String catalog,String shemaPattern,
        String tableNamPattern ,String types[]) throws SQLException
publi ReultSet getSchemas() throws SQLException
publi ReultSet getCatalogs() throws SQLException
publiynchronized ReultSet getTableTypes() throws SQLException
publiynchronized ReultSet getColumns(String catalog,String shemaPattern,
        String tableNamPattern ,String columnNamPattern) throws SQLException
publiynchronized ReultSet getColumnPriviges(String catalog, String shema,
        String table,String columnNamPattern) throws SQLException
publiynchronized ReultSet getTablePriviges(String catalog,String shemaPattern,
        String tableNamPattern) throws SQLException
publiynchronized ReultSet getBetRowIdentifir(String catalog,String shema,
        String table, int sope,  boolean nulable) throws SQLException
publiynchronized ReultSet getVersonColumns(String catalog,String shema,
        String table) throws SQLException
publiynchronized ReultSet getPriaryKeys(String catalog, String shema,
        String table) throws SQLException
publiynchronized ReultSet getImportedKeys(String catalog,String shema,
        String table) throws SQLException
publiynchronized ReultSet getExportedKeys(String catalog,String shema,
        String table) throws SQLException
publiynchronized ReultSet getCrossReference(String priaryCatalog,
        String priarySchema, String priaryTable, String foregnCatalog,
        String foregnSchema, String foregnTable) throws SQLException
publiynchronized ReultSet getTypeInfo() throws SQLException
publiynchronized ReultSet getIndexInfo(String catalog, String shema,
        String table, boolean unique,boolean approxiate) throws SQLException
```

从上面的方法可以看出，DatabasMetaData 对象给出了底层 DBMS 的功能和限制的信息。对应用程序员非常有用的一套信息包括描述数据库中表格结构细节的方法，以及描述表格名字、预先存储的过程名字等方法。

在 Java 应用程序中使用 DatabasMetaData 对象的一个例子是多层次可扩展应用程序的开发。如果底层数据库引擎支持某种特性，一个 Java 应用程序就可以查询它，如果不支持

则 Java 可以调用替代的方法来完成任务，这样，如果 DBMS 中不提供某种特性，Java 也不会失败。

11.3.3　ReultSetMetaData

ReultSetMetaData 类提供了对 ReultSet 结果集中列的类型和属性的访问方法。这些方法将作用于对结果集做一般性处理的程序，以确保在一个组织中各种应用程序的统一性，因为名字和大小是从数据库自身中取出的。下面列出了该类的基本方法：

publint getColumnCount() throws SQLException
返回结果集中列的数目；
publi booleanutoIncrent(int column) throws SQLException
指定列是否自动编码，因而只读；
publi booleanasSensitive(int column) throws SQLException
指定列是否字母区分大小写；
publi booleanSearchable(int column) throws SQLException
指定列是否可以用在查找语句中；
publi booleanurrency(int column) throws SQLException
指定列是否可以位金额型；
publint iNullable(int column) throws SQLException
指定列是否允许位 null 值；
publi booleanSigned(int column) throws SQLException
指定列中是否包含有符号数；
publint getColumnDiplaySize(int column) throws SQLException
返回指定列通常最大的字符长度；
publi String getColumnLabel(int column) throws SQLException
返回指定列的标题；
publi String getColumnNam(int column) throws SQLException
返回指定列的名称；
publi String getSchemaNam(int column) throws SQLException
publint getPreon(int column) throws SQLException
返回指定列的小数位数；
publint getScal(int column) throws SQLException
返回指定列小数点右面的数字；
publiynchronized String getTableNam(int column) throws SQLException
返回指定列的表名；
publi String getCatalogNam(int column) throws SQLException
返回指定列的表目录名；
publint getColumnType(int column) throws SQLException
返回指定列的 SQL 类型；
publi String getColumnTypeNam(int column) throws SQLException

返回指定列数据员的明确类型名；

publi booleanReadOnly(int column) throws SQLException

指定列是否只读；

publiynchronized booleanWritable(int column) throws SQLException

指定列是否可写；

publi booleanDefiniteyWritable(int column) throws SQLException

对指定列写是否会成功。

11.4 JDBC 语句

JDBC 支持三种类型的语句：

- Statent
- PreparedStatent
- CalableStatent

这三种 JDBC 类型语句的区别在于 SQL 语句准备和执行的时间不同。如果是简单的 Statent 对象，SQL 的准备和执行将同步进行；对于 PreparedStatent 对象，驱动程序存储执行计划以备以后来执行；而对于 CalableStatent 对象，SQL 语句实际上是调用一个已经优化的预先存储的过程。

下面将详细介绍这三种语句类型。

11.4.1 Statent

Statent 对象是在 Connetion 对象中利用 createStatent 方法创建的，其中最重要的方法是 executeQuery()、executeUpdate ()和 execute()。

- 当执行结果返回一个 ReultSet 对象时，可以采用 executeQuery()方法；
- 当执行结果返回多个 ReultSet 对象时，可以采用 execute()方法；
- 当执行结果不返回 ReultSet 对象，例如利用 update、delte 等创建的 SQL 语句，可以采用 exeuteUpdate ()方法。

下面列出了 Statent 对象中的所有方法：

publiynchronized ReultSet executeQuery(String sql) throws SQLException

执行 SQL 语句，并返回一个结果集，适用于 st 语句。

publiynchronized int executeUpdate(String sql) throws SQLException

执行 SQL 语句，但并不返回结果集，适用于 update、irt 、delte 等操作。

publiynchronized void cose() throws SQLException

关闭对象。

publint getMaxFidSize() throws SQLException

publiynchronized void stMaxFidSize(int maxFidSize) throws SQLException

publint getMaxRows() throws SQLException

publiynchronized void stMaxRows(int maxRows) throws SQLException
publiynchronized void stEsapeProceng(boolannable) throws SQLException
publint getQueryTiout() throws SQLException
publiynchronized void stQueryTiout(int sonds) throws SQLException
publiynchronized void cance() throws SQLException
publi SQLWarning getWarnings() throws SQLException
publiynchronized void carWarnings() throws SQLException
publiynchronized void stCursorNam(String cursorNam) throws SQLException
publiynchronized booleanxeute(String sql) throws SQLException
执行一个 SQL 语句。
publiynchronized ReultSet getReultSet() throws SQLException
返回一个结果集。
publiynchronized int getUpdateount() throws SQLException
publiynchronized boolean getMoreReults() throws SQLException

11.4.2　PreparedStatent

PreparedStatent　对象可以准备一个 SQL 语句，并将该 SQL 语句串传给底层 DBMS，但不执行这个 SQL 语句，而是可能返回一个 JDBC 驱动程序内部存储在 PreparedStatent 对象中的优化的执行计划的句柄。

同 Statent　对象不同，PreparedStatent　对象中的 exeuteQuery ()、executeUpdate()和 execute()方法不需要任何参数，它只是调用完成已经优化过的 SQL 语句。

PreparedStatent　对象的一个主要的特性是处理各种类型的参数，这些参数在 SQL 语句中是以“ ？ ”作为参数标记代替实际值来说明的，例如：

String sql="st * fro databasnamhere ID=?　　and nam=? ";

其中 databasnam　为数据库表名，ID 和 nam　为数据库中表的两个子段名，该 SQL 语句的作用就是需要查询 ID 和 nam　为指定内容的记录，而 ID 和 nam　的内容先用“ ？ ”来替代，由下面的语句来指定其内容。

指定的参数内容可以用 stXXX ()方法来和参数相关联，其中 XXX 为参数类型，例如整形为 int，字符串为 String 等，其中的参数索引由在 sql 语句中的顺序来决定，第一个“ ？ ”的索引为 1，第二个“ ？ ”的索引为 2 等。

PreparedStatent　对象中有如下的方法。

publiynchronized ReultSet executeQuery() throws SQLException
执行 SQL 语句，并返回一个结果集，适用于 st　　语句。
publiynchronized int executeUpdate() throws SQLException
执行 SQL 语句，但并不返回结果集，适用于 update、irt　、delte　等操作。
publiynchronized void stNull(int paramterIndex,int sqlType) throws SQLException
publiynchronized void stBoolan(int paramterIndex,boolan x) throws SQLException
publiynchronized void stByte(int paramterIndex,byte x) throws SQLException
publiynchronized void stShort(int paramterIndex,short x) throws SQLException

```
publiynchronized void stInt(it paramterIndex, int x) throws SQLException
publiynchronized void stLong(int paramterIndex, long x) throws SQLException
publiynchronized void stFloat(int paramterIndex,float x) throws SQLException
publiynchronized void stDouble(int paramterIndex,double x) throws SQLException
publiynchronized void stBigDeal(int paramterIndex,
        BigDeal x) throws SQLException
publiynchronized void stString(int paramterIndex,String x) throws SQLException
publiynchronized void stByte(int paramterIndex,byte x[]) throws SQLException
publiynchronized void stDate(int paramterIndex,Date x) throws SQLException
publiynchronized void stTi(int paramterIndex,Ti x) throws SQLException
publiynchronized void stTitamp(int paramterIndex,
        Titamp x) throws SQLException
publiynchronized void stAStream(int paramterIndex,
        InputStream x, int lngth) throws SQLException
publiynchronized void stUnicodeStream(int paramterIndex,
        InputStream x, int lngth) throws SQLException
publiynchronized void stBinaryStream(int paramterIndex,
        InputStream x, int lngth) throws SQLException
publiynchronized void carParamters() throws SQLException
publiynchronized void stObjet(int paramterIndex,Objet x,
        int targetSQLType, int sal) throws SQLException
publiynchronized void stObjet(int paramterIndex,
        Objet x, int targetSQLType) throws SQLException
publiynchronized void stObjet(int paramterIndex,Objet x) throws SQLException
设置不同的参数在指定的参数索引。
publiynchronized booleanxeute() throws SQLException
```

在使用 PreparedStatent 对象的时候，驱动程序实际上只发送执行计划 ID 和参数给 DBMS，这样做可以减少网络通信的负担，因此很适应 Internet 中的 Java 应用程序。或者当你在一个 Java 程序中需要多次执行一个 SQL 语句时，就最好使用 PreparedStatent 对象，它的执行效率比 Statent 对象更高一点。

11.4.3 CalableStatent

JDBC 允许通过 CalableStatent 类和转义字句字符串来使用预先存储的过程，而预先存储的过程集中完成商业逻辑处理和查询的运行。对于一个安全、一致和可管理的多层客户机/服务器系统来说，数据访问应该允许使用预先存储的过程。

一个 CalableStatent 对象是由 Connetion 对象中的 preparedCal()方法来创建的。PreparedCal() 方法需要一个字符串作为参数，这个字符串称为一个转义字句（eape cause ），它的形式为：

{[? =] Cal <stored procedure nam> { [<paramter1>, <paramter2>,　　……] }}

CalableStatent　　类支持参数，这些参数是从一个预先存储的过程中送出的 OUT 类型或者向一个预先存储过程传送值的 IN 类型。参数标记（用“ ？ ”来表示）必须用在返回值（如果有的话）或者任何输出参数的位置，因为参数标记是和预先储存过程中的程序变量绑定的，输入参数可以是常量或者是变元参数。对于一个动态参数的语句，转义字符串的形式为：

{[? =] Cal <stored procedure nam> { [<　？>, <？>, ……] }}

CalableStatent　　类是从 PreparedStatent　　类继承过来的，在调用 exeuteQuery() 、executeUpdate()和 execute 方法之前，OUT 参数应该用 registerOutparamter() 方法进行登记。CalableStatent　　类的方法如下：

```
publi void registerOutParamter(int paramterIndex,    int sqlType) throws SQLException
publi void registerOutParamter(int paramterIndex,
int sqlType, int sal) throws SQLException
publi booleanasNull() throws SQLException
publi String getString(int paramterIndex) throws SQLException
publi boolean getBoolean(int paramterIndex) throws SQLException
publi byte getByte(int paramterIndex) throws SQLException
publihort getShort(int paramterIndex) throws SQLException
publint getInt(int paramterIndex) throws SQLException
publiong getLong(int paramterIndex) throws SQLException
publi float getFloat(int paramterIndex) throws SQLException
publi doubl getDoubl(int paramterIndex) throws SQLException
publi BigDeal getBigDeal(int paramterIndex,int sal) throws SQLException
publi byte[] getByte(int paramterIndex) throws SQLException
publi Date getDate(int paramterIndex) throws SQLException
publi Ti getTi(int paramterIndex) throws SQLException
publi Titamp getTitamp(int paramterIndex) throws SQLException
publi Objet getObjet(int paramterIndex) throws SQLException
```

11.4.4　ReultSet 类

调用 StatentPrepared　、Statent　　或 CalableStatent　　的 executeQuery()方法都将返回一个 ReultSet　对象。ReultSet　对象实际上是一个管式的数据集合，即它是含有按统一的列组织的成行的数据。在 JDBC 中，Java 程序一次只能看到一行数据，在应用程序中只能利用 next()方法来得到数据的下一行，当得到一行数据后，在利用对象的索引或列的字段名来获取域值，实例如下所示。

```
publi void readData()
{
        try
```

```
            {
               Cas.forNam("sun.jdbc.odbcJdbcOdbcDriver");
                  //设置的 ODBC 数据源为“ tep    ”
               connetion=DriverManager.getConnetion("jdbc:odbc:tep");
                  //指定 SQL 语句
               String sql="st       nam, age, phone   from peopl";
                  //准备执行 sql 语句
               statent=connetion.prepareStatent(sql);
                     //执行并返回一个结果集
               ReultSet reult=   statent.executeQuery();
                  //利用 ReultSet  对象的 next（）函数来获取下一行数据，没有则返回 fal

                {
                     //利用索引来获取域值，即打印该行数据的 nam  字段值
                     Syste.out.println(reult.getString(1));
                     //利用列的子段名获取域值，即打印该行数据的 age 和 pho    字段值
                     Syste.out.println(reult.getInt(    "age"));
                     Syste.out.println(reult.getString(    "phone"))
                  }
                  reult.cose();
                  statent.cose();
                  conntion.cose();
               }
         }
         catch(Exception e)
         {
                  Syste.out.println(e.toString());
         }
   }
```

利用 ReultSet 对象可以对结果集进行控制和操作，ReultSet 类有下列方法。

publiynchronized booleanxt() throws SQLException
结果集指针下移执行下一个记录。
publiynchronized void cose() throws SQLException
关闭结果集。
publi booleanasNull() throws SQLException
判断结果集是否为空。
publiynchronized String getString(int column) throws SQLException
publiynchronized boolean getBoolan(int column) throws SQLException

(reult.next())

publiynchronized byte getByte(int column) throws SQLException
publiynchronized short getShort(int column) throws SQLException
publiynchronized int getInt(int column) throws SQLException
publiynchronized long getLong(int column) throws SQLException
publiynchronized float getFloat(int column) throws SQLException
publiynchronized doubl getDouble(int column) throws SQLException
publiynchronized BigDeal getBigDeal(int column,int sal) throws SQLException
publiynchronized byte[] getByte(int column) throws SQLException
publiynchronized Date getDate(int column) throws SQLException
publiynchronized Ti getTi(int column) throws SQLException
publiynchronized Titamp getTitamp(int column) throws SQLException
publiynchronized InputStream getAStream(it column) throws SQLException
publiynchronized InputStream getUnicodeStream(int column) throws SQLException
publiynchronized InputStream getBinaryStream(int column) throws SQLException
publiynchronized String getString(String columnNam) throws SQLException
publiynchronized boolean getBoolan(String columnNam) throws SQLException
publiynchronized byte getByte(String columnNam) throws SQLException
publiynchronized short getShort(String columnNam) throws SQLException
publiynchronized int getInt(String columnNam) throws SQLException
publiynchronized long getLong(String columnNam) throws SQLException
publiynchronized float getFloat(String columnNam) throws SQLException
publiynchronized doubl getDouble(String columnNam) throws SQLException
publiynchronized BigDeal getBigDeal(String columnNam,int sal) throws SQLException
publiynchronized byte[] getByte(String columnNam) throws SQLException
publiynchronized Date getDate(String columnNam) throws SQLException
publiynchronized Ti getTi(String columnNam) throws SQLException
publiynchronized Titamp getTitamp(String columnNam) throws SQLException
publiynchronized InputStream getAStream(String columnNam) throws SQLException
publiynchronized InputStream getUnicodeStream(String columnNam) throws SQLException
publiynchronized InputStream getBinaryStream(String columnNam) throws SQLException

得到结果集当前记录指定列索引或列名的各种不同类型的数据。

publiynchronized SQLWarning getWarnings() throws SQLException
publiynchronized void carWarnings() throws SQLException
publi String getCursorNam() throws SQLException
publiynchronized ReultSetMetaData getMetaData() throws SQLException
publiynchronized Objet getObjet(int column) throws SQLException
publiynchronized Objet getObjet(String columnNam) throws SQLException

publiynchronized int findColumn(String columnNam) throws SQLException
查找指定的列名，并返回该列的索引。

11.5 访问数据库实例

下面的例子中我们将利用 Java 来实现对数据库中的记录进行添加、删除和修改操作。设表的结构如名为 peopl 的表 11-2 所示。

表 11-2 pepl 表结构定义

ID	nam	age		ph	
1	张三	34	男	8657458	浙江
2	李四	41	男	5698524	北京

下面详细说明数据库开发该例的过程。

11.5.1 设计主窗口

在 Jbuider2 中利用向导创建一个新的工程 databas.jpr 和一个主窗体类 MainFram.java ，并在 MainFram.java 类的 desgn 视图中加入 GridControl 和四个按钮“ 添加 ”、“ 删除 ”、“ 修改 ”和“ 退出 ”，并进行适当的布局，激活按钮的响应事件和 GridControl 的 subfocusChanged 事件，同时修改 GridConrol 控件的标题保持同数据库中一致（“ 姓名 ”、“ 性别 ”、“ 年龄 ”、“ 电话 ”、“ 备注 ”）。最后得到 MainFram.java 如源程序清单 11-1 所示。

[*源程序清单 11-1*]

```
package databas;

iport java.awt.*;
iport java.awt.event.*;
iport borland.jbcl.layout.*;
iport borland.jbclontrol.*;
iport java.sql.*;
iport java.uti*;
iport borland.jbclode*;

publias MainFramxtends DeoratedFram
{
   GridControl gridControl1 = ne    GridControl();
   Button button1 = ne    Button();
   Button button2 = ne    Button();
   Button button3 = ne    Button();
```