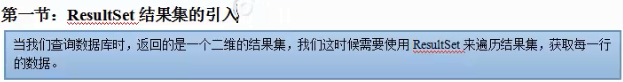
利用ResultSet接口进行查询Query

1. **ResultSet结果集**的引入:



1. **获取ResultSet对象：**

**利用PreparedStatement的executeQuery（）方法：返回一个ResultSet结果集。**



注意：**这里的sql语句使select 语句**。

1. ResultSet中的方法主要是getXxx方法和updateXxx方法，这两种方法最多，除此之外，还有对光标控制和位置判断的方法。
2. 对于**ResultSet结果集**中的光标移动及位置判断有关的方法是：

处在最前头与指向第一行不一样，同样，处在最后头与指向最后一行也不是一回事。

1. **beforeFirst、afterLast方法**：移动最前或最后，返回布尔值。

**isBeforeFirst、isAfterLast方法**：判断是否是在最前头或者最末端。

1. **first、last方法**:移动到第一个或最后一个，返回布尔值；

**isFirst、isLast方法**：判断是不是出在第一个或最后一个。

1. **next、previous方法**：向下或上移动一个，返回布尔值。

注意：**ResultSet**的光标位置初始在第一行的前面，第一次调用next方法，会使光标处于第一行，第二次调用next方法会使光标处于第二行。故在获取结果时，首先调用一次next方法。

如：**while（rs.next()）{ code….}**

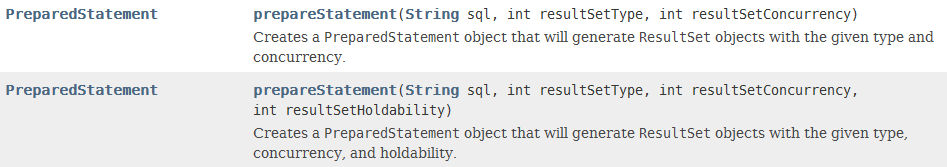
1. **absolute（int row）方法**：让光标指**向指定的行**，返回布尔值。
2. **relative（int row）方法：让光标上下移动row个位置，row可0可负可正。**

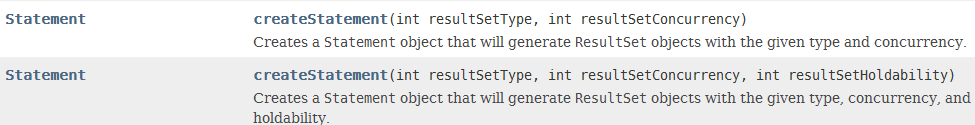


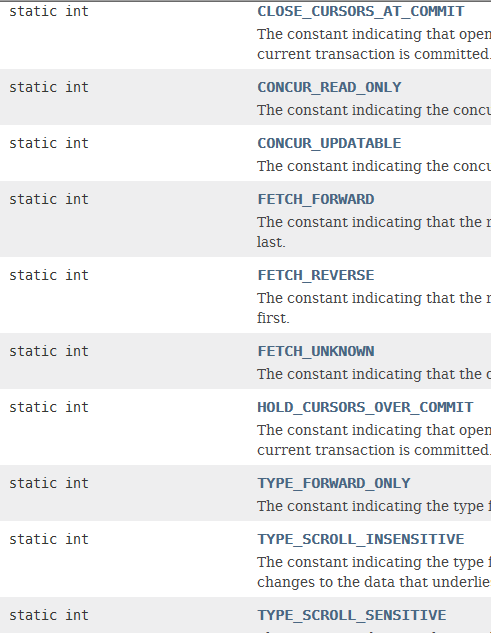
1. **moveToCurrentRow和moveToInsertRow：**
2. ResultSet具有一些属性，用于指定结果集是否可以滚动：

如只可以向前或只可以向后滚动。

在Connection中创建Statement或PreparedStatement的时候，可以传入Resultset的属性，从而对Resultset的滚动特性进行限制。







1. 使用**ResultSet**遍历查询结果：

可以利用**ResultSet结果集**对表格中的所有的数据进行遍历，也可以只对某些数据进行遍历。对应的sql语句为：

**String sql = "select \* from new\_table";//查询所有的数据**

**为了提高代码的可读性，一般把所有的列名列出了，顺序可以与表中的顺序不一致，便于后面利用getXxx时通过类名获取数据；getXxx方法中参数的列号必须与select语句后的列名顺序一致。同样为了提高代码的可读性，getXxx中的参数也要使用类名字符串。**

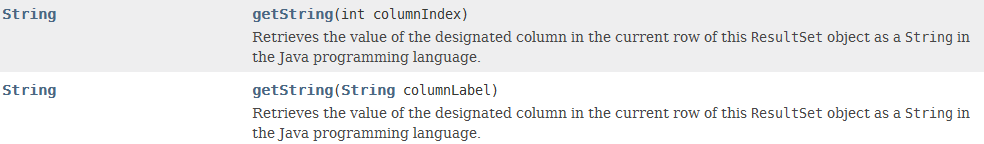
**String sql = "select id from new\_table";//只查询id号**

**String sql = "select id,name from new\_table where id<10";//只查询某些行的某些列，利用where进行限制条件。**

1. **ResultSet接口**中的getXxx（）方法：

有两种等价的形式：方法的小括号中，可以利用**列号**表示（**列号是select语句后面列名的顺序，不是数据库表中的列号**），也可以利用**列名字符串**表示出来，两种方法等价，为了提高代码可读性，建议使用**列名字符串**。

如**getString（int index）和getString（String label）方法**：



1. 分页查询：

有些数据库不支持分页查询，MySQL数据库支持分页查询，SQL语句是：

**select \* from users limit 100,20 //利用limit关键词，100表示从第100个开始，20表示查询的个数。**

Connection conn = DriverUtil.getConnection();

String sql = "select \* from users limit 100,20";

PreparedStatement psm = conn.prepareStatement(sql);

ResultSet rs = psm.executeQuery();

while(rs.next()) {

System.out.println(rs.getInt(1));

}

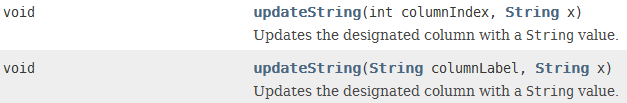
DriverUtil.close(rs,psm,conn);

1. ResultSet的**判断是否关闭**的方法： **isClosed方法，返回布尔值。**



1. **ResultSet结果集的updateXxx（）方法**：

**也是即可以利用列序号，也可以利用列标签（列名）。**



1. 利用**ResultSet结果集**进行**查询**：

（1）利用**列索引**进行查询：

**String sql = "select \* from t\_book";**

PreparedStatement psm = con.prepareStatement(sql);

**ResultSet rs = psm.executeQuery();**

while(rs.next()) {

int id = rs.getInt(1);

String bookName = rs.getString(2);

float price = rs.getFloat(3);

String author = rs.getString(4);

int bookId = rs.getInt(5);

StringBuilder sb = new StringBuilder();

sb.append(id);sb.append(bookName);

sb.append(price);sb.append(author);sb.append(bookId);

System.out.println(sb.toString());

}

du.close(con,psm);

（2）利用**列标签**即列名进行**查询**：**这样的可读性比较好**。

**int** id = rs.getInt("id");

String bookName = rs.getString("bookName");

**float** price = rs.getFloat("price");

String author = rs.getString("author");

**int** bookId = rs.getInt("bookTypeId");

1. ResultSet还可以具有**更新功能**：

需要在创建**PreparedStatement或Statement对象**的时候，传入参数。

这种功能使用的较少，因为查询里再**更新数据**，会造成混乱，如果需要更新就用update语句。

**PreparedStatement psm = conn.prepareStatement(sql,**

**ResultSet.TYPE\_SCROLL\_SENSITIVE, ResultSet.CONCUR\_UPDATABLE);**

**利用ResultSet的updateXxx方法：进行更新操作，最后，执行updateRow（）。**

**ResultSet.TYPE\_SCROLL\_SENSITIVE表示可以感知数据库的变化，而ResultSet.TYPE\_SCROLL\_INSENSITIVE表示不可以感知数据库的变化。**

**如：**Connection conn = DriverUtil.getConnection();

String sql = "select \* from users limit 60,20";

PreparedStatement psm = conn.prepareStatement(sql,

ResultSet.TYPE\_SCROLL\_SENSITIVE, ResultSet.CONCUR\_UPDATABLE);

ResultSet rs = psm.executeQuery();

while (rs.next()) {

**rs.updateString("name", "SBBBSSSSSS");**

**rs.updateRow();**

}

DriverUtil.close(rs, psm, conn);

1. 把**查询的结果封装成对象**，然后存储到**List集合**中。

List<Book> bookList = new ArrayList<Book>();

Connection con = du.getConnection();

String sql = "**select \* from t\_book**";

PreparedStatement psm = con.prepareStatement(sql);

ResultSet rs = psm.executeQuery();

while(rs.next()) {

Book book = new Book(rs.getInt("id"),rs.getString("bookName"),rs.getInt("price"),rs.getString("author"),rs.getInt("bookTypeId"));

bookList.add(book);

}

System.out.println(bookList.toString());

for(Book book : bookList) {

System.out.println(book.toString());

}