
**博客：** https://www.cnblogs.com/HOsystem/p/14116443.html

# 2、具体内容

既然java.sql包里面提供有Statement接口可以实现数据库的操作，那么为什么又要提供有一个PreparedStatement接口实现数据库的操作呢？

## ■Statement接口操作问题

下面就以更新的操作为例，在Statement接口里面如果要想执行SQL语句，那么一定要通过字符串实现SQL结构的定义，但是这种定义如果要结合到用户输入数据的情况下就有可能会有问题存在了，下面通过一个程序做一个简单的模拟。

**范例：分析Statement接口操作问题**

```
" INSERT INTO news(nid,title,read,price,content,pubdate) VALUES "
                        + " (news_seq.nextval,'MLDN新闻'老李写的' ,99,99.8,
                        +  '这个春天有点冷' , TO_DATE('1971-04-10','yyyy-mm-dd'));
```

```
package cn.mldn.demo;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
public class JDBCDemo {
    private static final String DATABASE_DRVIER = "oracle.jdbc.driver.OracleDriver" ;
    private static final String DATABASE_URL = "jdbc:oracle:thin:@localhost:1521:mldn" ;
    private static final String DATABASE_USER = "scott" ;
    private static final String DATABASE_PASSWORD = "tiger" ;
    public static void main(String[] args) throws Exception {
        String title = "MLDN新闻'老李写的" ;      // 问题
        int read = 99 ;
        double price = 99.8 ;
        String content = "这个春天有点冷" ;
        String pubdate = "1971-04-10" ;   // 问题一：日期使用了字符串描述
        String sql = " INSERT INTO news(nid,title,read,price,content,pubdate) VALUES "
                + " (news_seq.nextval,'"+title+"',"+read+","+price+", "
```

```
                    + " '"+content+"', TO_DATE('"+pubdate+"','yyyy-mm-dd'))" ; // 问题二：维护困难
            System.out.println(sql);
            Connection conn = null ;  // 每一个Connection接口对象描述的就是一个用户连接
            Class.forName(DATABASE_DRVIER) ; // 向容器之中加载数据库驱动程序
            conn = DriverManager.getConnection(DATABASE_URL, DATABASE_USER, DATABASE_PASSWORD) ;
            Statement stmt = conn.createStatement() ; // 创建数据库的操作对象
            int count = stmt.executeUpdate(sql) ; // 返回影响的行数
            System.out.println("更新操作影响的数据行数：" + count);
            conn.close();  // 数据库的连接资源有限一定要关闭
        }
}
```

利用Statement执行的SQL语句问题有如下三种：

·不能很好的描述出日期的形式；

·需要进行SQL语句的拼凑处理，而导致的结果就是：SQL语句的编写与维护困难；

·对于一个写免肝的字符数据无法进行合理拼凑；

所以，现在就可以发现，虽然Statement可以操作数据库，但是其在操作的过程之中并不是那么的方便，而它最大的弊端：需要进行SQL语句的拼凑。

# ■PreparedStatement操作数据库

为了解决Statement接口存在的SQL执行问题，所以在java.sql包里面又提供有一个Statement子接口：PreparedStatement，这个接口最大的好处是可以编写正常的SQL(数据不再和SQL语法混合在一起)，同时利用占位符的形式，在SQL正常执行完毕后可以进行数据的设置。观察PreparedStatement接口定义：

```
public interface PreparedStatement extends Statement
```

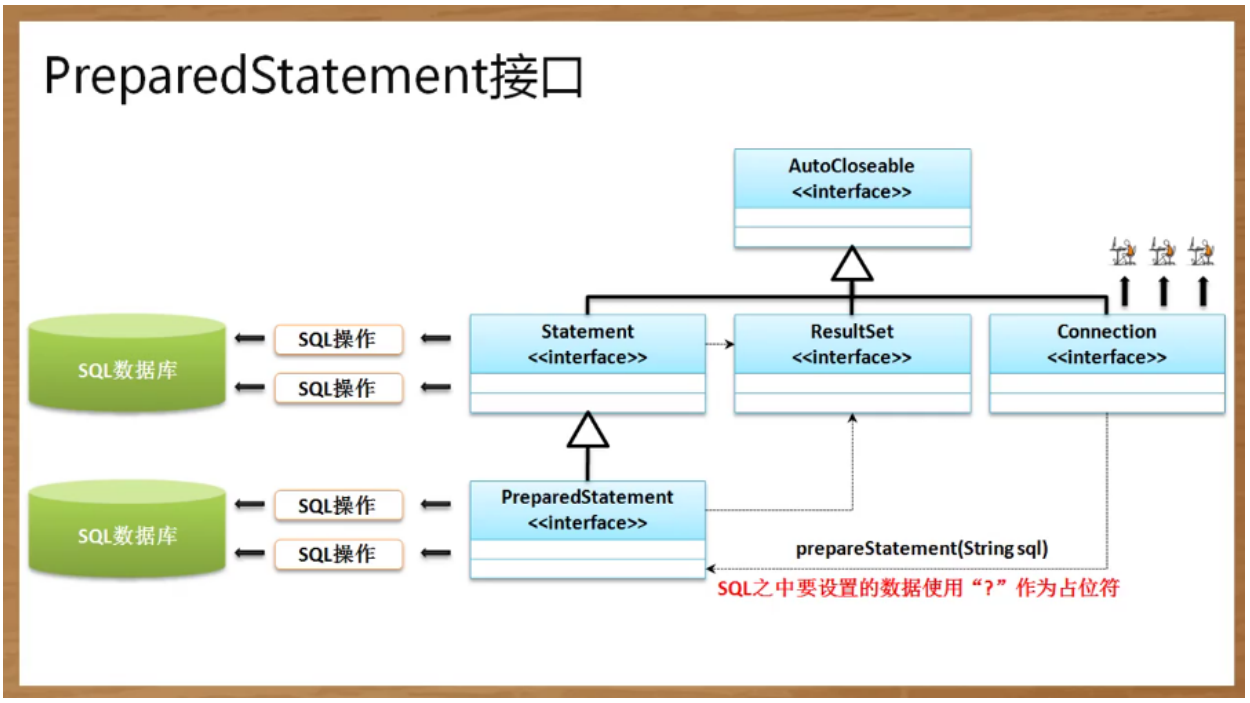
如果要想获取PreparedStatement接口的实例,则依然需要通过Connection接口实现,创建方法：

·创建PreparedStatement接口对象：public PreparedStatement prepareStatement(String sql) throws SQLException；

由于SQL语句已经在创建PreparedStatement接口对象的时候提供了，所以在执行数据库操作的时候也要更换方法。

·数据库更新：public int executeUpdate() throws SQLException；

·数据库查询：public ResultSet executeQuery() throws SQLException；

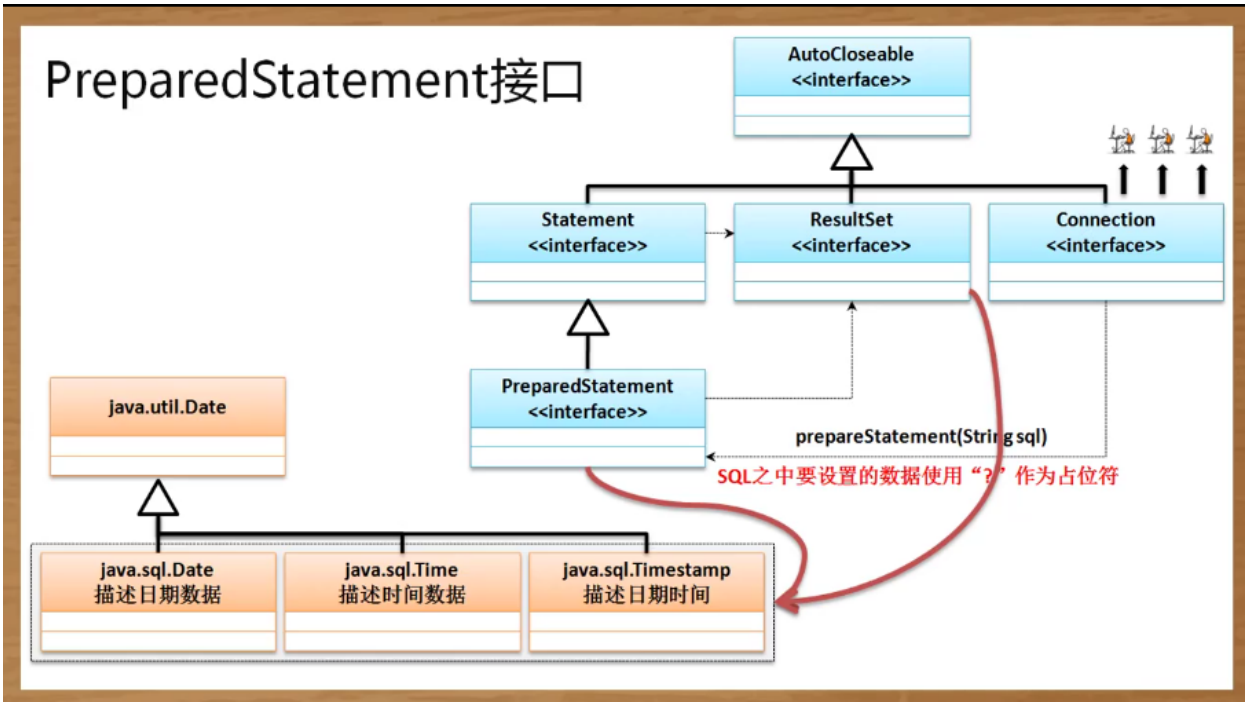



**范例：利用PreparedStatement解决之前的数据操作问题**

```
package cn.mldn.demo;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.util.Date;
public class JDBCDemo {
	private static final String DATABASE_DRVIER = "oracle.jdbc.driver.OracleDriver" ;
	private static final String DATABASE_URL = "jdbc:oracle:thin:@localhost:1521:mldn" ;
	private static final String DATABASE_USER = "scott" ;
	private static final String DATABASE_PASSWORD = "tiger" ;
	public static void main(String[] args) throws Exception {
		String title = "MLDN新闻'老李写的" ;
		int read = 99 ;
		double price = 99.8 ;
		String content = "这个春天有点冷" ;
		Date pubdate = new Date() ;
		String sql = " INSERT INTO news(nid,title,read,price,content,pubdate) VALUES "
				+ " (news_seq.nextval,?,?,?,?,?)" ; // 使用了 “?” 作为占位符
		Connection conn = null ;  // 每一个Connection接口对象描述的就是一个用户连接
		Class.forName(DATABASE_DRVIER) ; // 向容器之中加载数据库驱动程序
		conn = DriverManager.getConnection(DATABASE_URL, DATABASE_USER,
DATABASE_PASSWORD) ;
		PreparedStatement pstmt = conn.prepareStatement(sql) ; // 创建数据库的操作对象
		// 在执行具体的数据库操作之前需要为占位符设置内容，按照顺序设置
		pstmt.setString(1, title);
		pstmt.setInt(2, read);
		pstmt.setDouble(3, price);
		pstmt.setString(4, content);
		pstmt.setDate(5, new java.sql.Date(pubdate.getTime()));
		int count = pstmt.executeUpdate() ; // 返回影响的行数
		System.out.println("更新操作影响的数据行数：" + count);
		conn.close();  // 数据库的连接资源有限一定要关闭
```

```
    }
}
```

在JDBC里面不管使用但是PreparedStatement设置的日期时间还是使用ResultSet获取的日期时间实际上都是java.util.Date的子类，也就是说现在是如下的对应关系：




---

# ■使用PreparedStatement实现数据查询

清楚了PreparedStatement实现更新处理之后，那么下面可以使用其实现数据的查询处理操作，由于在开发之中PreparedStatement的使用是最广泛的，下面将列举几个有代表性的查询。

1、查询全部数据：

```
package cn.mldn.demo;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.Date;
public class JDBCDemo {
    private static final String DATABASE_DRVIER = "oracle.jdbc.driver.OracleDriver" ;
    private static final String DATABASE_URL = "jdbc:oracle:thin:@localhost:1521:mldn" ;
    private static final String DATABASE_USER = "scott" ;
    private static final String DATABASE_PASSWORD = "tiger" ;
    public static void main(String[] args) throws Exception {
        String sql = "SELECT nid,title,read,price,content,pubdate FROM news " ;
        Connection conn = null ;  // 每一个Connection接口对象描述的就是一个用户连接
        Class.forName(DATABASE_DRVIER) ; // 向容器之中加载数据库驱动程序
        conn = DriverManager.getConnection(DATABASE_URL, DATABASE_USER,
DATABASE_PASSWORD) ;
```

```
            PreparedStatement pstmt = conn.prepareStatement(sql) ; // 创建数据库的操作对象
            // 在执行具体的数据库操作之前需要为占位符设置内容，按照顺序设置
            ResultSet rs = pstmt.executeQuery() ;
            while (rs.next()) {     // 现在如果发现还有数据行未输出
                  int nid = rs.getInt(1) ;
                  String title = rs.getString(2) ;
                  int read = rs.getInt(3) ;
                  double price = rs.getDouble(4) ;
                  String content = rs.getString(5) ;
                  Date pubdate = rs.getDate(6) ;
                  System.out.println(nid + "、" + title + "、" + read + "、" + price + "、" +
content + "、" + pubdate);
            }
            conn.close();  // 数据库的连接资源有限一定要关闭
      }
}
```

2、根据id进行数据查询

```
package cn.mldn.demo;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.Date;
public class JDBCDemo {
      private static final String DATABASE_DRVIER = "oracle.jdbc.driver.OracleDriver" ;
      private static final String DATABASE_URL = "jdbc:oracle:thin:@localhost:1521:mldn" ;
      private static final String DATABASE_USER = "scott" ;
      private static final String DATABASE_PASSWORD = "tiger" ;
      public static void main(String[] args) throws Exception {
            String sql = "SELECT nid,title,read,price,content,pubdate FROM news WHERE
nid=?" ;
            Connection conn = null ;  // 每一个Connection接口对象描述的就是一个用户连接
            Class.forName(DATABASE_DRVIER) ; // 向容器之中加载数据库驱动程序
            conn = DriverManager.getConnection(DATABASE_URL, DATABASE_USER,
DATABASE_PASSWORD) ;
            PreparedStatement pstmt = conn.prepareStatement(sql) ; // 创建数据库的操作对象
            pstmt.setInt(1, 5); // 设置nid的数据
            // 在执行具体的数据库操作之前需要为占位符设置内容，按照顺序设置
            ResultSet rs = pstmt.executeQuery() ;
            while (rs.next()) {     // 现在如果发现还有数据行未输出
                  int nid = rs.getInt(1) ;
                  String title = rs.getString(2) ;
                  int read = rs.getInt(3) ;
                  double price = rs.getDouble(4) ;
                  String content = rs.getString(5) ;
                  Date pubdate = rs.getDate(6) ;
                  System.out.println(nid + "、" + title + "、" + read + "、" + price + "、" +
content + "、" + pubdate);
            }
            conn.close();  // 数据库的连接资源有限一定要关闭
      }
```

```
}
```

3、在进行全部数据查询的时候如果返回的内容过多则一定会造成内存的大量占用，那么此时可以使用分页的形式实现数据的查询处理(模糊查询)

```
package cn.mldn.demo;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.Date;
public class JDBCDemo {
	private static final String DATABASE_DRVIER = "oracle.jdbc.driver.OracleDriver" ;
	private static final String DATABASE_URL = "jdbc:oracle:thin:@localhost:1521:mldn" ;
	private static final String DATABASE_USER = "scott" ;
	private static final String DATABASE_PASSWORD = "tiger" ;
	public static void main(String[] args) throws Exception {
		int currentPage = 2 ; // 当前页
		int lineSize = 5 ; // 每页显示的数据行
		String column = "title" ; // 模糊查询列
		String keyWord = "MLDN" ; // 查询关键字
		String sql = "SELECT * FROM ( "
				+ " SELECT nid,title,read,price,content,pubdate,ROWNUM rn "
				+ " FROM news WHERE " + column + " LIKE ? AND ROWNUM<=?
ORDER BY nid) temp "
				+ " WHERE temp.rn>?" ;
		Connection conn = null ;  // 每一个Connection接口对象描述的就是一个用户连接
		Class.forName(DATABASE_DRVIER) ; // 向容器之中加载数据库驱动程序
		conn = DriverManager.getConnection(DATABASE_URL, DATABASE_USER,
DATABASE_PASSWORD) ;
		PreparedStatement pstmt = conn.prepareStatement(sql) ; // 创建数据库的操作对象
		pstmt.setString(1, "%" + keyWord + "%");
		pstmt.setInt(2, currentPage * lineSize);
		pstmt.setInt(3, (currentPage - 1) * lineSize);
		// 在执行具体的数据库操作之前需要为占位符设置内容，按照顺序设置
		ResultSet rs = pstmt.executeQuery() ;
		while (rs.next()) {     // 现在如果发现还有数据行未输出
			int nid = rs.getInt(1) ;
			String title = rs.getString(2) ;
			int read = rs.getInt(3) ;
			double price = rs.getDouble(4) ;
			String content = rs.getString(5) ;
			Date pubdate = rs.getDate(6) ;
			System.out.println(nid + "、" + title + "、" + read + "、" + price + "、" +
content + "、" + pubdate);
		}
		conn.close();  // 数据库的连接资源有限一定要关闭
	}
}
```

4、统计指定关键字标题的新闻数量

```
package cn.mldn.demo;
import java.sql.Connection;
```

```
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
public class JDBCDemo {
	private static final String DATABASE_DRVIER = "oracle.jdbc.driver.OracleDriver" ;
	private static final String DATABASE_URL = "jdbc:oracle:thin:@localhost:1521:mldn" ;
	private static final String DATABASE_USER = "scott" ;
	private static final String DATABASE_PASSWORD = "tiger" ;
	public static void main(String[] args) throws Exception {
		String column = "title" ; // 模糊查询列
		String keyWord = "MLDN" ; // 查询关键字
		String sql = "SELECT COUNT(*) FROM news WHERE " + column + " LIKE ?" ;
		Connection conn = null ;  // 每一个Connection接口对象描述的就是一个用户连接
		Class.forName(DATABASE_DRVIER) ; // 向容器之中加载数据库驱动程序
		conn = DriverManager.getConnection(DATABASE_URL, DATABASE_USER,
DATABASE_PASSWORD) ;
		PreparedStatement pstmt = conn.prepareStatement(sql) ; // 创建数据库的操作对象
		pstmt.setString(1, "%" + keyWord + "%");
		// 在执行具体的数据库操作之前需要为占位符设置内容，按照顺序设置
		ResultSet rs = pstmt.executeQuery() ;
		if (rs.next()) {
			long count = rs.getLong(1) ;
			System.out.println("复合条件的数据量：" + count);
		}
		conn.close();  // 数据库的连接资源有限一定要关闭
	}
}
```

在使用COUNT()函数做统计查询的时候一定会返回查询结果，如果表中没有数据则返回0.