实验九 JDBC进阶（3）

一、相关知识点

1. JDBC基本概念
2. 批处理
3. DAO和OR映射

二、实验目的：

掌握批量SQL语句执行的方法，理解DAO和ORM的基本概念

三、实验内容：

1、 利用批量SQL语句执行的方法实现实验5中的最后两个方法：

A、编写批量借阅读书函数： public void lendbooks(String readerId,Collection barcodes) .... 。其中第二个参数为图书条码集合。

```
1  public void lendbooks(String readerId,Collection<String> barcodes) throws
   BaseException {
2      Object[] objects = barcodes.toArray();
3      for (int i = 0; i < objects.length; i++) {
4          lend((String) objects[i],readerId);
5      }
6      BeanReader r = (new ReaderManager()).loadReader(readerId);
7      if (r == null) throw new BusinessException("读者不存在");
8      if (r.getRemoveDate() != null) throw new BusinessException("读者已注
   销");
9      if (r.getStopDate() != null) throw new BusinessException("读者已挂失");
10     for (int i = 0; i < objects.length; i++) {
11         BeanBook book = (new BookManager()).loadBook((String) objects[i]);
```

```
            if (book == null) throw new BusinessException("图书不存在");
            if (!"在库".equals(book.getState())) throw new BusinessException("图书"
+ book.getState());
        }
        List<BeanBook> lentbooks = this.loadReaderLentBooks(readerId);
        if (r.getLendBookLimitted() <= lentbooks.size()) {
            throw new BusinessException("超出限额");
        }
        Connection conn = null;
        try {
            conn = DBUtil.getConnection();
            conn.setAutoCommit(false);
            String sql = "insert into
BeanBookLendRecord(readerid,bookBarcode,lendDate,lendOperUserid,penalS
um) values(?,?,?,?,0)";
            java.sql.PreparedStatement pst = conn.prepareStatement(sql);
            pst.setString(1, readerId);
            //手动事务
            conn.setAutoCommit(false);
            int count = 0;
            for (int i = 0; i < objects.length; i++) {
                BeanBook book = (new BookManager()).loadBook((String)
objects[i]);
                if (book == null) throw new BusinessException("图书不存在");
                if (!"在库".equals(book.getState())) throw new BusinessException("图
书" + book.getState());
                pst.setString(2, (String) objects[i]);
                pst.setTimestamp(3, new
java.sql.Timestamp(System.currentTimeMillis()));
                pst.setString(4, SystemUserManager.currentUser.getUserid());
                pst.addBatch();
                count++;
                sql = "update BeanBook set state='已借出' where barcode=?";
                pst = conn.prepareStatement(sql);
                pst.setString(1, (String) objects[i]);
                pst.addBatch();
                count++;
                if(count>=25000) {
                    //每25000条数据进行一次批量插入操作
                    pst.executeBatch();
                    pst.clearBatch();
                    conn.commit();
```

```
                count = 0;
            }
        }
        if(count != 0)
        {
            pst.executeBatch();
            pst.clearBatch();
            conn.commit();
            count = 0;
        }
    } catch (SQLException e) {
        e.printStackTrace();
        throw new DbException(e);
    } finally {
        if (conn != null)
            try {
                conn.rollback();
                conn.close();
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
    }
}
```

B、编写批量设置罚金函数：public void setPenalSum(String readerId,Map<String,Double> penalSums) .... 。其中第二个参数的key为barcode，value为改读者尚未归还图书的罚金（注意，不要设置已经归还图书的罚金）。

```
public void setPenalSum(String readerId,Map<String,Double> penalSums) throws Exception {
    Connection conn = null;
    java.sql.PreparedStatement pst = null;
    try {
        conn = DBUtil.getConnection();
        String sql = "update beanbooklendrecord set penalSum = ? where readerid = ? and bookBarcode = ? and returnDate is null";
        pst = conn.prepareStatement(sql);
        pst.setObject(2,readerId);
        //手动事务
        conn.setAutoCommit(false);
        int count = 0;
```

```
            for (Map.Entry<String,Double> x : penalSums.entrySet()) {
                String barcode = x.getKey();
                double penal = x.getValue();
                pst.setObject(1,penal);
                pst.setObject(3,barcode);
                pst.addBatch();
                count++;
                if(count>=25000) {
                    //每25000条数据进行一次批量插入操作
                    pst.executeBatch();
                    pst.clearBatch();
                    conn.commit();
                    count = 0;
                }
            }
            if(count != 0)
            {
                pst.executeBatch();
                pst.clearBatch();
                conn.commit();
                count = 0;
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            DBUtil.closeResorce(conn,pst);
        }

    }
```

2、 模仿SystemUserDAO类，实现BookDAO类，并改造BookManager类，使之通过BookDAO操作数据库。

【实验结果与分析】

A、 给出BookDAO类代码。

```
package cn.edu.zucc.booklib.dao;

import cn.edu.zucc.booklib.model.BeanBook;
import cn.edu.zucc.booklib.util.BaseException;
import cn.edu.zucc.booklib.util.BusinessException;
import cn.edu.zucc.booklib.util.DBUtil;
```

```
7  import cn.edu.zucc.booklib.util.DbException;
8
9  import java.sql.Connection;
10 import java.sql.PreparedStatement;
11 import java.sql.ResultSet;
12 import java.sql.SQLException;
13 import java.util.ArrayList;
14 import java.util.List;
15
16 public class BookDAO {
17     public List<BeanBook> searchBook(String keyword, String bookState) throws
   BaseException {
18         List<BeanBook> result = new ArrayList<BeanBook>();
19         Connection conn = null;
20         try {
21             conn = DBUtil.getConnection();
22             String sql = "select * from view_book" + " where state='" + bookState + "' ";
23             if (keyword != null && !"".equals(keyword))
24                 sql += " and (bookname like ? or barcode like ?)";
25             sql += " order by barcode";
26             java.sql.PreparedStatement pst = conn.prepareStatement(sql);
27             if (keyword != null && !"".equals(keyword)) {
28                 pst.setString(1, "%" + keyword + "%");
29                 pst.setString(2, "%" + keyword + "%");
30
31             }
32
33             java.sql.ResultSet rs = pst.executeQuery();
34             while (rs.next()) {
35                 BeanBook b = new BeanBook();
36                 b.setBarcode(rs.getString(1));
37                 b.setBookname(rs.getString(2));
38                 b.setPubid(rs.getString(3));
39                 b.setPrice(rs.getDouble(4));
40                 b.setState(rs.getString(5));
41                 b.setPubName(rs.getString(6));
42                 result.add(b);
43             }
44         } catch (SQLException e) {
45             e.printStackTrace();
46             throw new DbException(e);
47         } finally {
```

```
48              if (conn != null)
49                  try {
50                      conn.close();
51                  } catch (SQLException e) {
52                      // TODO Auto-generated catch block
53                      e.printStackTrace();
54                  }
55          }
56          return result;
57
58      }
59
60      public void createBook(BeanBook b) throws BaseException {
61
62
63          if (b.getBarcode() == null || "".equals(b.getBarcode()) ||
    b.getBarcode().length() > 20) {
64              throw new BusinessException("条码必须是1-20个字");
65          }
66          if (b.getBookname() == null || "".equals(b.getBookname()) ||
    b.getBookname().length() > 50) {
67              throw new BusinessException("图书名称必须是1-50个字");
68          }
69          Connection conn = null;
70          try {
71              conn = DBUtil.getConnection();
72              String sql = "select * from BeanBook where barcode=?";
73              PreparedStatement pst = conn.prepareStatement(sql);
74              pst.setString(1, b.getBarcode());
75              ResultSet rs = pst.executeQuery();
76              if (rs.next()) throw new BusinessException("条码已经被占用");
77              rs.close();
78              pst.close();
79
80              sql = "insert into BeanBook(barcode,bookname,pubid,price,state)
    values(?,?,?,?,'在库')";
81              PreparedStatement ps = conn.prepareStatement(sql);
82              ps.setObject(1, b.getBarcode());
83              ps.setObject(2, b.getBookname());
84              ps.setString(3, b.getPubid());
85              ps.setDouble(4, b.getPrice());
86
```

```
87
88              ps.execute();
89              pst.close();
90          } catch (SQLException e) {
91              e.printStackTrace();
92              throw new DbException(e);
93          } finally {
94              if (conn != null)
95                  try {
96                      conn.close();
97                  } catch (SQLException e) {
98                      // TODO Auto-generated catch block
99                      e.printStackTrace();
100                 }
101         }
102
103     }
104
105     public void modifyBook(BeanBook b) throws BaseException {
106         if (b.getBookname() == null || "".equals(b.getBookname()) ||
    b.getBookname().length() > 50) {
107             throw new BusinessException("图书名称必须是1-50个字");
108         }
109         Connection conn = null;
110         try {
111             conn = DBUtil.getConnection();
112             String sql = "select * from BeanBook where barcode=?";
113             java.sql.PreparedStatement pst = conn.prepareStatement(sql);
114             pst.setString(1, b.getBarcode());
115             java.sql.ResultSet rs = pst.executeQuery();
116             if (!rs.next()) throw new BusinessException("图书不存在");
117             rs.close();
118             pst.close();
119             sql = "update BeanBook set bookname=?,pubid=?,price=?,state=? where
    barcode=?";
120             pst = conn.prepareStatement(sql);
121             pst.setString(1, b.getBookname());
122             pst.setString(2, b.getPubid());
123             pst.setDouble(3, b.getPrice());
124             pst.setString(4, b.getState());
125             pst.setString(5, b.getBarcode());
126             pst.execute();
```

```
127             pst.close();
128         } catch (SQLException e) {
129             e.printStackTrace();
130             throw new DbException(e);
131         } finally {
132             if (conn != null)
133                 try {
134                     conn.close();
135                 } catch (SQLException e) {
136                     // TODO Auto-generated catch block
137                     e.printStackTrace();
138                 }
139         }
140     }
141 
142     public BeanBook loadBook(String barcode) throws DbException {
143         Connection conn = null;
144         try {
145             conn = DBUtil.getConnection();
146             String sql = "select * from view_book " + " where barcode=? ";
147             java.sql.PreparedStatement pst = conn.prepareStatement(sql);
148             pst.setString(1, barcode);
149             java.sql.ResultSet rs = pst.executeQuery();
150             if (rs.next()) {
151                 BeanBook b = new BeanBook();
152                 b.setBarcode(rs.getString(1));
153                 b.setBookname(rs.getString(2));
154                 b.setPubid(rs.getString(3));
155                 b.setPrice(rs.getDouble(4));
156                 b.setState(rs.getString(5));
157                 b.setPubName(rs.getString(6));
158                 return b;
159             }
160         } catch (SQLException e) {
161             e.printStackTrace();
162             throw new DbException(e);
163         } finally {
164             if (conn != null)
165                 try {
166                     conn.close();
167                 } catch (SQLException e) {
168                     // TODO Auto-generated catch block
```

```
                e.printStackTrace();
            }
        }
        return null;
    }
}
```

B、 给出改造后BookManager类的各个方法的代码。

```
public List<BeanBook> searchBook(String keyword, String bookState) throws BaseException, SQLException {
    Connection conn = null;
    try {
        conn = DBUtil.getConnection();
        return dao.searchBook(keyword, bookState);
    } catch (SQLException e) {
        e.printStackTrace();
    } catch (BaseException e) {
        e.printStackTrace();
    } finally {
        if (conn != null) {
            try {
                conn.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

    return null;

}


public void createBook(BeanBook b) throws BaseException, SQLException {
    Connection conn = null;
    try {
        conn = DBUtil.getConnection();

        dao.createBook(b);
    } catch (SQLException e) {
        e.printStackTrace();
```

```
33      } catch (BaseException e) {
34          e.printStackTrace();
35      } finally {
36          if (conn != null) {
37              try {
38                  conn.close();
39              } catch (SQLException e) {
40                  e.printStackTrace();
41              }
42          }
43
44      }
45
46  }
47
48
49  public void modifyBook(BeanBook b) throws BaseException, SQLException {
50      Connection conn = null;
51      try {
52          conn = DBUtil.getConnection();
53
54          dao.modifyBook(b);
55      } catch (SQLException e) {
56          e.printStackTrace();
57      } catch (BaseException e) {
58          e.printStackTrace();
59      } finally {
60          if (conn != null) {
61              try {
62                  conn.close();
63              } catch (SQLException e) {
64                  e.printStackTrace();
65              }
66          }
67      }
68
69  }
70
71
72  public BeanBook loadBook(String barcode) throws DbException, SQLException {
73      Connection conn = null;
74      try {
```

```
        conn = DBUtil.getConnection();

        dao.loadBook(barcode);
    } catch (SQLException e) {
        e.printStackTrace();
    } catch (DbException e) {
        e.printStackTrace();
    } finally {
        if (conn != null) {
            try {
                conn.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

    return null;
```

3、 在BaseDAO中，增加方法，实现根据主码提取对象的方法load。

【实验结果与分析】

A、 写出函数代码。

```
public T load(Connection conn, Class<T> clazz, Map<String,Object> primaryKey) {
    PreparedStatement ps = null;
    ResultSet rs = null;
    try {
        //primayKey的key为主码的各个属性名称，value为主码值
        //假设对象的类名和表名一致，属性名和字段名一致

        String tableName = clazz.getSimpleName();

        String sql = "select * from "+tableName+" where"; //动态构建sql
        for (Map.Entry mp : primaryKey.entrySet()) {
            sql += " " + mp.getKey().toString() + " = ? and";
        }
        sql = sql.substring(0,sql.length()-3);

        Object[] parmas = new Object[primaryKey.size()];
        //从map中获取参数值，并写入params
```

```
        int pos = 0;
        for (Map.Entry<String, Object> mp : primaryKey.entrySet()) {
            parmas[pos++] = mp.getValue();
        }

        ps = conn.prepareStatement(sql);
        for (int i = 0; i < parmas.length; i++) {
            ps.setObject(i+1,parmas[i]);
        }
        rs = ps.executeQuery();
        ResultSetMetaData rsmd = rs.getMetaData();
        if (rs.next()) {
            T t = clazz.getDeclaredConstructor().newInstance();

            for (int i = 0; i < rsmd.getColumnCount(); i++) {
                //获取当前rs指针的i+1字段的值
                Object value = rs.getObject(i+1);

                //获取当前rs指针的i+1字段的列名
                String columnLable = rsmd.getColumnLabel(i+1);

                //将value赋值给对应属性，反射
                Field field = clazz.getDeclaredField(columnLable);
                field.setAccessible(true);
                field.set(t, value);
            }
            return t;
        }
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        DBUtil.closeResorce(null,ps,rs);
    }

    return null;
```