和分布式计算。Java是通过applet以及使用Java Web Start来进行客户端编程，利用可运行于任何Web服务器的JSP技术和功能强大、可移植性好Servlet技术可以方便地构建系统查询B/S、调案B/S应用。

5.2 系统组成类设计及实现

本节通过类图、运行界面等描述了RFID房地产档案管理系统实现，详细描述了RFID房地产档案管理系统实体管理系统部分公共类、子系统专门类的设计实现。授权用户通过初始登陆界面（如图5-1）登陆特定功能的界面，完成具体功能操作，系统主要的功能有入库、出库、转架、盘点等。本节中所涉及的详细代码参见附录二[30]。

5.2.1 公共类设计实现

RFID实体档案管理系统开发中的公共类包括“数据库访问”类、“标签类”、“标签信息格式”类、“数据库批量更新”类等[31]。

1) 数据库访问类(DatabaseAccess)：RFID档案管理系统各功能的实现始终离不开数据库的支持，“数据库访问”类设计如图5-1所示，用该类实现与数据库的连接、数据库查询、更新等操作（部分代码见附录2)。

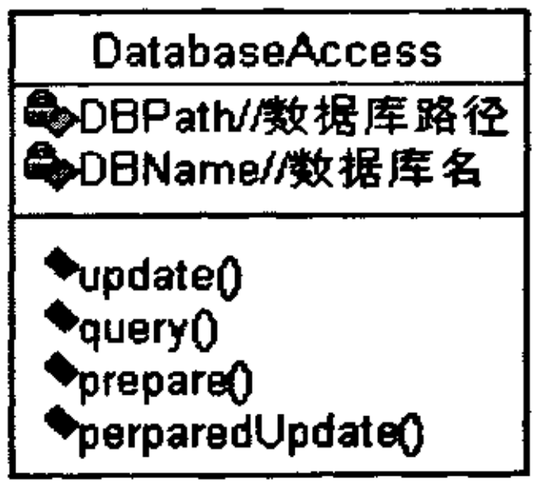

图5-1 数据库访问类

2) 标签类及标签数据过滤标准类：标签类主要用于标签整体性的管理，标签数据过滤标准类对不同数据格式的标签进行统一的管理，将来如若数据格式变动则不影响系统的正常运行。标签类及标签数据过滤标准类设计参见图5-2所示，read和write分别完成标签数据的读取、写入处理，“标签”类通过输入的标签格式参数实现与读写器沟通，控制读写器读取标签中数据或向标签写入数据；数据过滤标准类中getFiltrateFile方法根据特定系统功能的编号首先获取包含数据块定义的XML文件，getStandard方法利用对相关的过滤XML文件的解

析后可以得到特定功能下读写标签的数据过滤标准（部分代码见附录2）。

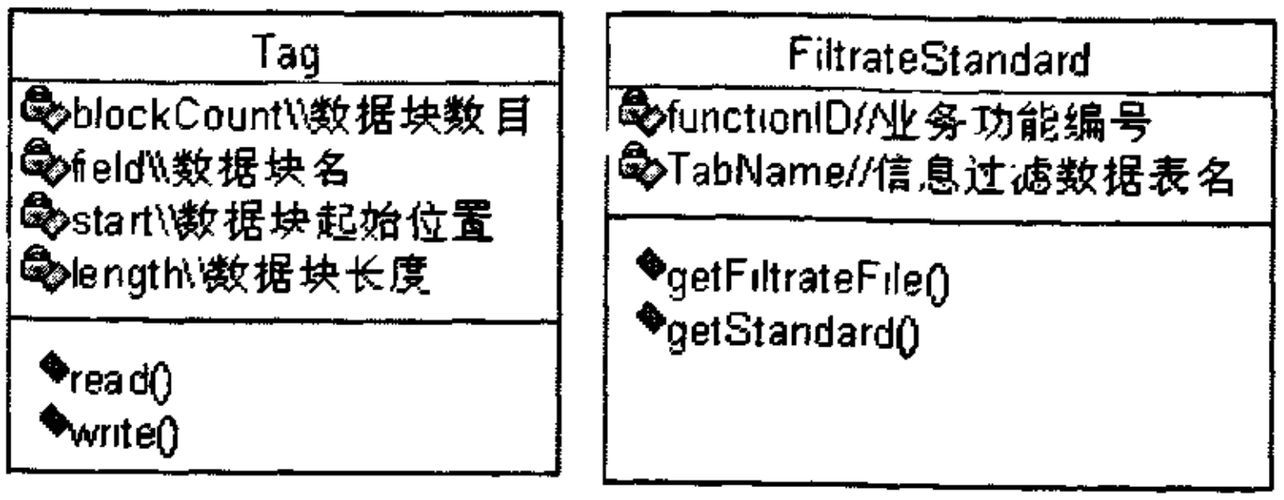

图5-2 标签类及标签数据过滤标准类

3）数据库批量更新类（BatchUpdateDB）及标签集类：基于RFID的档案管理系统在各功能处理完毕后都要进行相应的数据库的更新，如果在某一功能下在每次信息处理完毕后都进行独立的数据库更新，那么必须在批量标签处理完毕前始终保持着对特定数据库的连接，或者必须在每次更新时都连接数据库一次，研究发现，在利用SUN公司所提供的基本JDBC API进行数据库连接及操作时，当连接数或并发连接数过多时，系统内存资源会被大量消耗，没能及时释放的资源将最终造成系统运行效率大大降低，因此为了保证RFID系统运行效率，在本论文中在进行标签的批量处理时，数据库的更新我们亦采取批量处理的方式，建立数据库批量更新（BatchUpdateDB）即在批量标签尚未全部处理完毕前将需要更新的数据内容逐一添加到相应类型的XML临时文件，待批量表情全部处理完毕后完成由系统连接一次数据库实现数据的批量更新。标签集(TagSet)用于批量处理标签，数据库批量更新类（BatchUpdateDB）、标签集（TagSet）如图5-3所示：

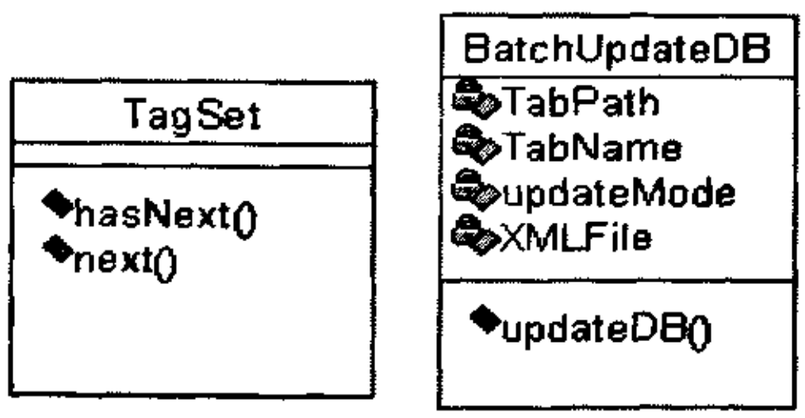

图5-3 数据库批量更新类及标签集类

4）标签验证类（TagVerify），在RFID中我们只对合法的电子标签进行数据处理，因此在系统中每一功能实现前我们都要对电子标签的合法性加以验证，通过建立标签验证类我们实现对电子标签的合法性验证（标签验证类图如图5-4下

所示)。

图 5-4 标签验证类

5.2.2 子系统专门类的设计实现

RFID 档案管理系统的子系统包括实体签收子系统、入库子系统、出库子系统、盘点子系统、转架子系统[32,33]等。以下对各子系统的专用类进行设计:

1) 实体签收子系统

在档案实体签收签我们要进行实体的签收验证,实体的签收验证模块功能在原来的条形码系统中已实现,在新的 RFID 系统中我们可以直接利用该模块的功能。在 RFID 系统中我们主要实现的是与条形码系统签收验证模块的集成及签收模块的数据库更新、电子标签信息更新,execute 方法用于执行签收命令,execute 方法将通过 comTabName 属性所指的签收数据表完成对批量数据信息的自动签收验证处理,并相应地生成与“数据库更新”类相关的各种 XML 临时文件内容。在所有签收电子标签信息处理完成后,execute 方法将利用生成的更新档案签收电子标签数据表 XML 文件和更新在库档案数据表 XML 文件更新签收电子标签数据表和档案签收数据表。用于执行档案的入库,其中数据库等的更新均在此操作中完成。同时调用条形码系统新创建接口的相应方法完成条形码系统所属相关数据库的更新,至此,实体入库处理完毕。具体设计类图如 5-5、5-6 下所示:

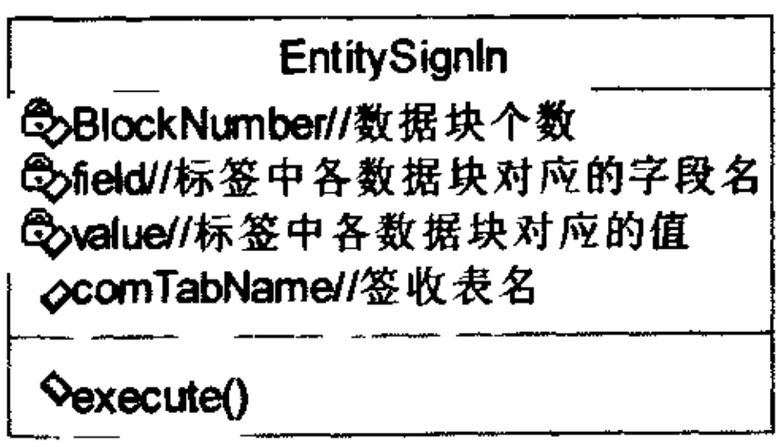

图 5-5 实体签收类

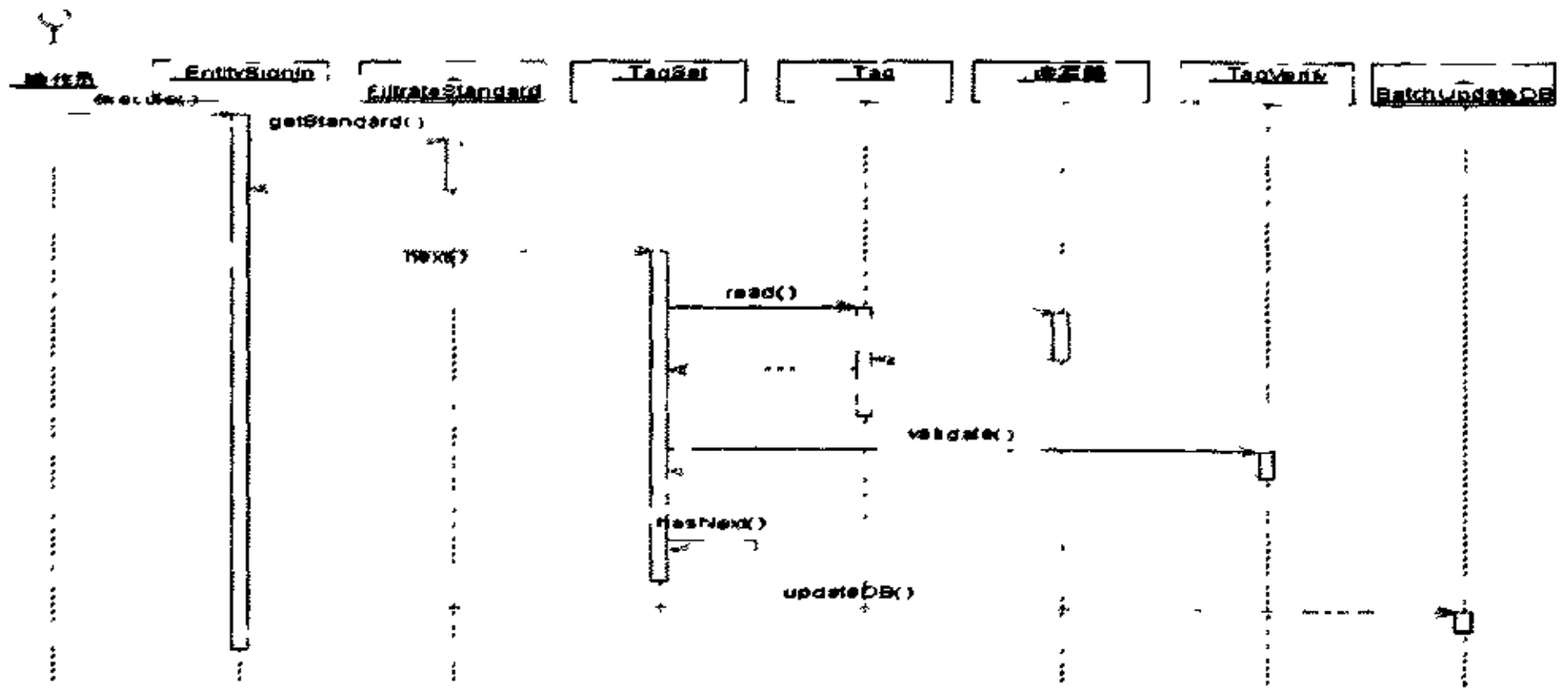

图 5-6　实体签收顺序图

2)　档案实体入库子系统

档案实体入库设计到新档案及旧档案入库两种情况[34]。通过建立EntityCheckIn 类进行档案入库的管理（见图 5-7 所示），其中 isNew 方法用于判断是新档案入库，还是旧档案入库，对于新档案利用 create 方法生成归档号；execute 方法将通过 comTabName 属性所指的入库数据表完成对批量数据信息的自动入库验证处理，并相应地生成与“数据库更新”类相关的各种 XML 临时文件内容。在所有入库电子标签信息处理完成后，execute 方法将利用生成的更新档案入库电子标签数据表 XML 文件和更新在库档案数据表 XML 文件更新入库电子标签数据表和在库档案数据表。用于执行档案的入库，其中数据库等的更新均在此操作中完成。同时调用条形码系统新创建接口的相应方法完成条形码系统所属相关数据库的更新，至此，实体入库处理完毕。

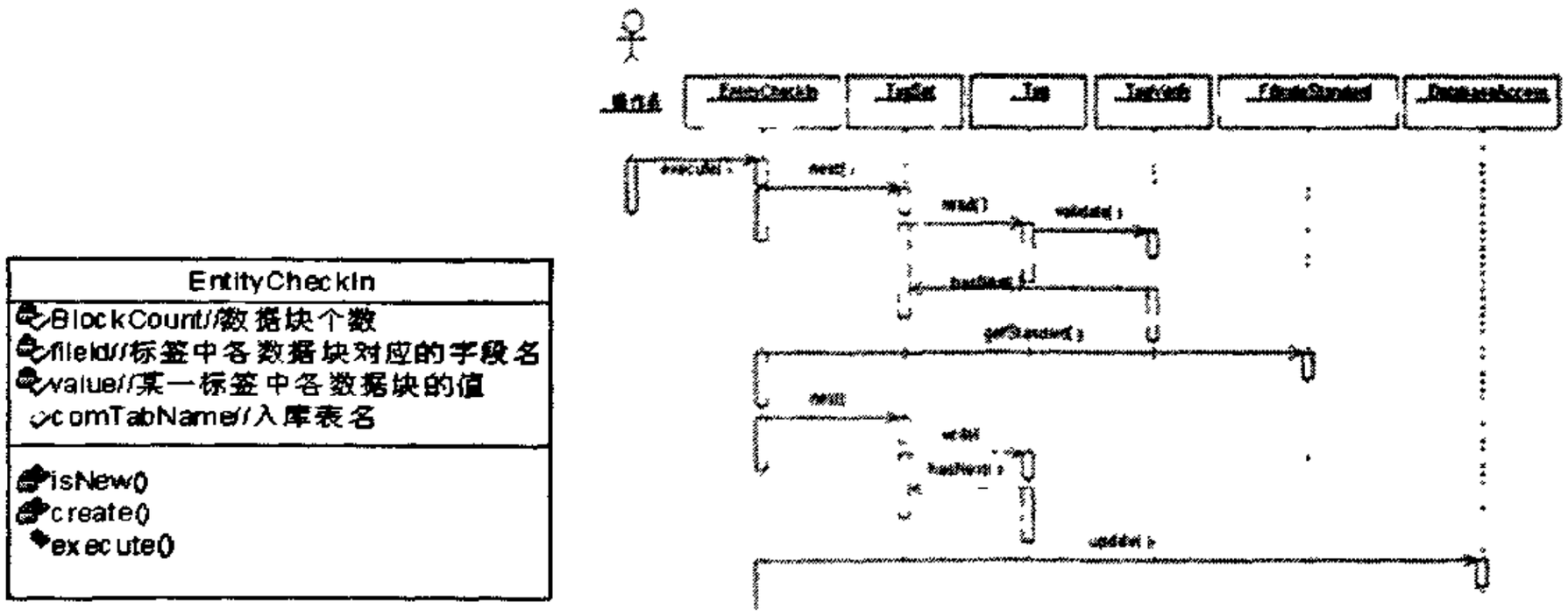

图 5-7 实体入库类（左）及实体入库序列图（右）

3)　档案实体出库、转架子系统

档案实体出库子系统专门类包括“实体出库”，“非法出库报警”类，实体转

架子系统包括“实体转架”类，出库、转架类均可参照入库子系统专门类设计。限于篇幅，此处具体设计略。

4) 档案盘点子系统

盘点子系统用专门的“盘点”类调用“标签过滤标准”类、“标签”类、“标签集”类、“数据库访问”类、“数据库批量更新”类等类的相关方法完成对在库档案书的统计、档案存放位置的校验及相关数据库的更新。限于篇幅，具体类的设计略。

5.3 标签过滤标准文件管理

使用了关系模式与 XML 文件相结合的方式建立的标签过滤标准文件可以有效地解决、预防电子标签中数据格式改变的问题。故如何能够有效地读取、建立标签过滤标准文件是我们所必须要重视的问题，以下我们详细描述了对标签过滤标准文件的管理实现。

5.3.1 相关技术—DOM 解析器介绍

DOM 是用与平台和语言无关的方式表示 XML 文档的官方 W3C 标准。DOM 是以层次结构组织的节点或信息片断的集合。这个层次结构允许开发人员在树中寻找特定信息。分析该结构通常需要加载整个文档和构造层次结构，然后才能做任何工作。由于它是基于信息层次的，因而 DOM 被认为是基于树或基于对象的。DOM 以及广义的基于树的处理具有几个优点。首先，由于树在内存中是持久的，因此可以修改它以便应用程序能对数据和结构做出更改。它还可以在任何时候在树中上下导航，而不是像 SAX 那样是一次性的处理。DOM 使用起来也要简单得多。

DOM 解析器把 XML 文档转化为一个包含其内容的树，并可以对树进行遍历。用 DOM 解析模型的优点是编程容易，开发人员只需要调用建树的指令，然后利用 navigation APIs 访问所需的树节点来完成任务。可以很容易的添加和修改树中的元素。

基于 DOM 解析器的优越性，结合本系统的特点，在论文中采用 DOM 解析器来实现标签过滤标准文件的管理。

5.3.2 标签过滤标准文件管理实现

本论文中通过建立 XMLTagManager 类来管理实现 XML 标签标准过滤文件的