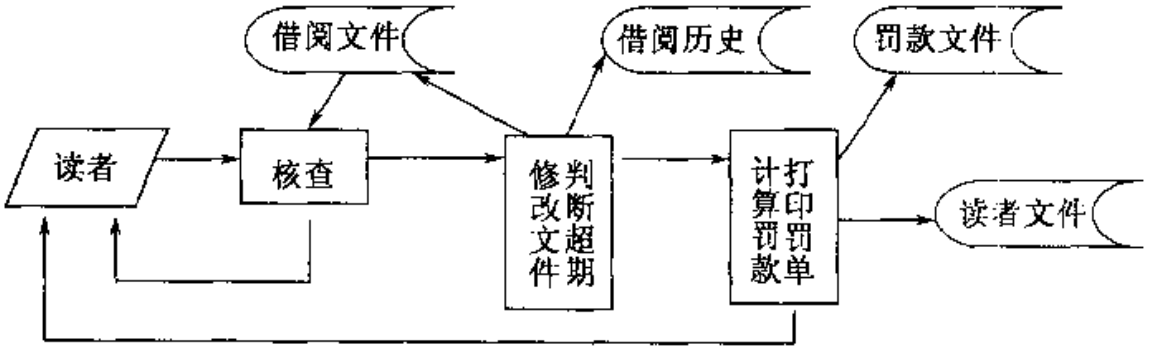

图 4-3　还书过程数据流图

（4）为了对图书管理系统做完整的描述，还需要对上面得到的逻辑模型做一些补充。首先采用图形的方式描述图书管理系统的用户界面，这样做的目的是保证整个系统的用户界面的一致性，同时也有助于后续的开发人员更好地理解系统需要实现的功能。在这就不罗列用户界面，以免与后面内容重复。其次，说明图书管理系统的一些特殊性能要求，如借书、还书服务花费的时间一次不得大于 5 分钟等。

前面着重对借还书流程进行了详细的阐述，以说明如何利用数据流图这一工具进行软件的分析，下面介绍图书管理系统的总体功能要求。简单的图书管理系统主要包括下面的功能：

- 借书处理：完成读者借书这一业务流程。
- 还书处理：完成读者还书这一业务流程。
- 罚款处理：解决读者借书超期的罚款处理。
- 新书上架：输入新书资料。
- 旧书淘汰：删除图书资料。
- 读者查询：根据读者号，查询读者借阅情况。

4.2　图书管理系统的数据分析

通过对图书系统的分析，可以得出该系统涉及四个实体：读者、图书、工作人员。通过对各实体数据关系的整理，我们可以画出如下 ER 图（图 4-4）。

这些实体涉及的数据项有：

读者：读者条码，姓名，身份证号，最多借书数，止借标志

图书：图书条码，书名，作者，出版社，出版日期，数量，停借标志

工作人员：工作人员ID，姓名，身份证号，密码职务

实体之间的联系涉及的数据项有：

借阅文件：读者条码，图书条码，借出日期，归还日期，操作人员ID

罚款文件：读者条码，罚款天数，罚款数，罚款日期，解止日期，操作人员ID

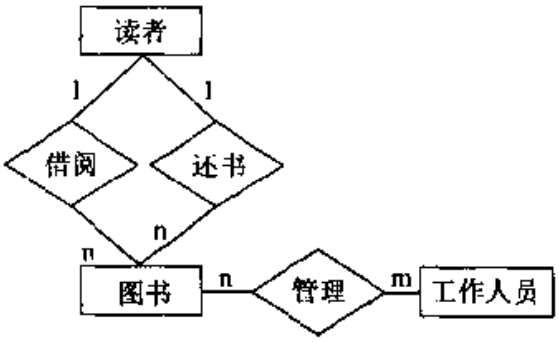

图 4-4 ER 图

如果将上述实体分别对应一个表，可以完成要实现的功能。但注意到在前面的分析中，强调要考虑处理借书、还书的效率。在上面的表结构中，不难发现随着借阅记录的逐渐增多，借阅文件的查询效率会降低，势必影响还书处理的效率，因而建议将表结构改为如下形式：

读者文件：读者条码，姓名，身份证号，最多借书数，止借标志

图书文件：图书条码，书名，作者，出版社，出版日期，停借标志

工作人员：工作人员 ID，姓名，身份证号，密码，职务

借阅文件：读者条码，图书条码，借出日期，操作人员 ID

借阅历史文件：读者条码，图书条码，借出日期，归还日期，借书操作人员 ID，还书操作人员 ID

罚款文件：读者条码，罚款天数，罚款数，罚款日期，操作人员ID

罚款历史文件：读者条码，罚款天数，罚款数，罚款日期，解止日期（解止日期指解除该读者止借标志的日期）

小　结

在这一章里，我们通过图书管理系统这一实例介绍了结构化系统分析的方法和使用 ER 图设计数据库的方法。

结构化系统分析和设计的方法主要是采用数据流程图等工具对整个系统的数据流进行描述。在使用 ER 图设计数据库结构时，主要任务是能将实体抽象出来，把握实体之间的联系。

第五章　图书管理系统的模块划分

根据前面对需求的分析，我们可以得到如图 5-1 的模块结构图。

下面我们对模块结构图中，最细的功能进行详细介绍。

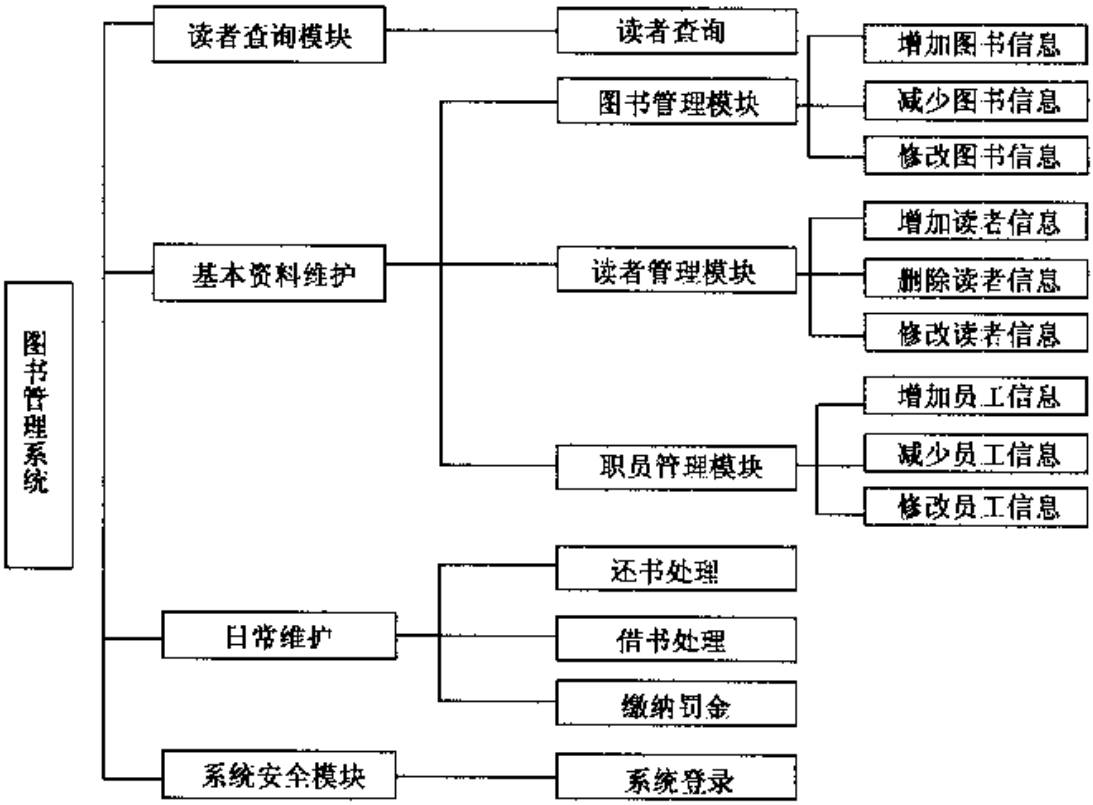

图 5-1　模块结构图

图书管理系统需要实现的功能主要有四大块：基本资料维护、日常维护、系统安全模块和读者查询模块。其中日常维护和基本资料维护是整个系统的核心。日常维护包括借书处理、还书处理和缴纳罚金。基本资料维护包括对读者、员工和图书等信息的维护，主要有读者信息的增删改，对员工资料进行增删改和对图书资料进行增删改。系统安全模块只是实现了最简单的系统登录检查。读者查询也只实现了简单的查询功能。下面对具体的模块做具体的介绍。

借书处理的主要功能是扫描读者条形码，扫描图书条形码，在数据库中插入一条借书记录，该记录包括读者条形码、图书条形码、借出日期。

还书处理的主要功能是扫描图书条形码，在借阅文件中找到相应的记录，

将该记录的相应数据插入到还书记录中，同时将借书记录删除。

缴纳罚金的主要功能是输入读者条形码，显示该读者的姓名、罚款金额和过期天数，如果读者交纳了罚金，则将读者文件的允许借阅标志置为“Y”。删除罚款文件中该读者对应的记录，将这一条记录同时插入到罚款历史文件中。

读者查询：允许读者根据自己的条形码或姓名查询自己的借书记录。

图书资料维护的功能包括输入新书资料、删除旧书资料，修改图书资料等。

读者资料维护的功能包括读者信息的输入、修改和删除。

工作人员信息维护主要包括工作人员信息的输入、修改和删除。

注销读者：将读者记录置止借标志，同时提供删除读者信息的功能。（这一部分在系统的原型中并没有实现）。

系统登录：是对用户名和输入的密码进行检查，已确定登录用户是否合法。用户名和密码的维护是在工作人员资料维护模块中实现的。

从上面对整个系统的模块的介绍，读者可以看到，本书中所要介绍的图书管理系统是一个非常简单的系统原型，这也是本书的目的所在，通过简单的例子，让读者对 C/S 结构的应用系统的开发有一定的了解。

第六章　图书管理系统的具体实现

6.1　建立数据库

前面已经根据系统的需求信息拟订了表结构，从这一节开始，将介绍系统的具体实现。使用的后台数据库服务器为 Access。本书所使用的所有代码均在以下操作环境下运行通过。

操作系统：Windows 2000

数据库服务器：Microsoft Access 2000

开发工具：Visual C++ 6.0

该图书管理系统所需的库结构为：

读者文件（reader）：

读者条码(reader_id)，姓名(name)，身份证号(IDCARD)，最多借书数(maxnum_can_borrow)，止借标志(flag_borrow)

图书文件(book)：

图书条码(book_id)，书名 book_nam，作者(author，出版社(press)，出版日期(press_date)，停借标志(flag_borrow)

职工文件（clerk）：

职员 ID(Clerk_ID)，姓名（name），身份证号（ID_Card），口令（Password）职务（office）

借阅文件(borrow)：

读者条码(reader_ID)，图书条码(book_id)，借出日期(borrow_date) 操作人员代码（B_Clerk_ID），ID

借阅历史文件(history)：

读者条码(reader_id)，图书条码(book_id)，借出日期(borrow_date)，归还日期(return_date)，借书操作人员代码（B_Clerk_ID），还书操作人员代码（R_Clerk_ID），ID

罚款文件(fine)：

读者条码(reader_id)，罚款天数(days)，罚款数(amount)，罚款日期(finedate)，操作人员(Clerk_ID)，ID

罚款历史文件(fine_history):

读者条码(reader_id)，罚款天数(days)，罚款数(amount)，罚款日期(fine_date) 解止日期(ok_date)，ID

一般在设计数据库结构时，我们常常在每个表上，尤其是一些存放业务流程数据的表上（如借阅文件、罚款文件等）增加一个对用户来说没有意义的字段，这个字段一般是长整型，并设其为自动加 1 的 identity 类型。这个字段主要是为了在业务流程数据表中惟一标识一条记录。

建立数据库的方式有很多种，一种是直接使用 Access 的图形界面建立相应的数据库，这种方式的好处是操作简单，但是不适用于大型系统的开发。第二种方式是使用辅助数据库设计的工具直接建立数据库，修改数据库，如 powerdesigner 等。建议使用这种方式，但需要开发人员熟悉这些工具。图 6-1 是使用 powerdesigner 设计的图书管理系统的数据库结构图。第三种方式是写 SQL 语句，将整个系统涉及的数据表，视图、触发器等用 SQL 语句写在一个.sql 的文本文件中，再用数据库支持的批量处理 SQL 语句的命令执行这个文件。这种方式在应用系统开发时用得比较普遍。在程序 6-1-1 中将看到图书管理系统的 SQL 语句。

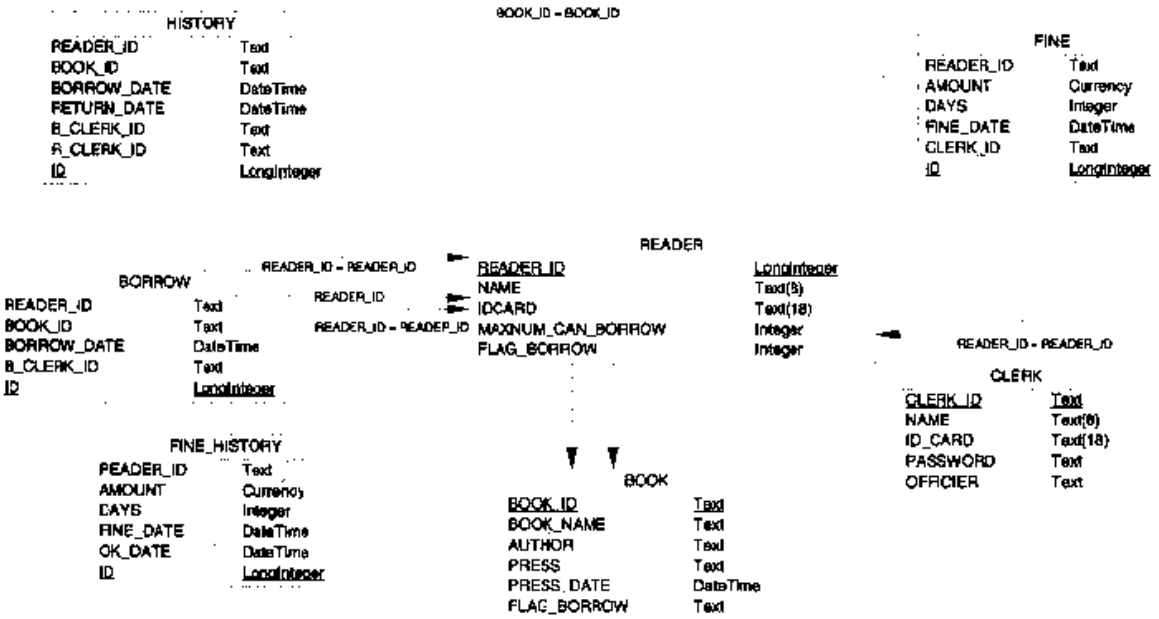

图 6-1 数据库结构图

程序6-1-1:

```
/* ============================================================ */
/*   Database name:  MODEL_1                                    */
/*   DBMS name:      Microsoft SQL Server 7.x                   */
```

```
/*   Created on:     02-1-23  21:29                                */
/* ============================================================== */

/* ============================================================== */
/*   Table: CLERK                                                 */
/* ============================================================== */
create table CLERK
(
   CLERK_ID          char(30)            not null,
   NAME              char(8)             null    ,
   ID_CARD           char(18)            null    ,
   PASSWORD          char(30)            null    ,
   OFFICIER          char(30)            null
)
go

/* ============================================================== */
/*   Table: READER                                                */
/* ============================================================== */
create table READER
(
   READER_ID         int                 not null,
   NAME              char(8)             null    ,
   IDCARD            char(18)            null    ,
   MAXNUM_CAN_BORROW  int                 null    ,
   FLAG_BORROW       int                 null
)
go

/* ============================================================== */
/*   Table: BOOK                                                  */
/* ============================================================== */
create table BOOK
(
```

```
    BOOK_ID          char(30)          not null,
    BOOK_NAME        char(50)          null    ,
    AUTHOR           char(30)          null    ,
    PRESS            char(50)          null    ,
    PRESS_DATE       datetime          null    ,
    FLAG_BORROW      char(10)          null
)
go

/* ============================================================ */
/*   Table: HISTORY                                             */
/* ============================================================ */
create table HISTORY
(
    READER_ID        char(30)          null    ,
    BOOK_ID          char(30)          null    ,
    BORROW_DATE      datetime          null    ,
    RETURN_DATE      datetime          null    ,
    B_CLERK_ID       char(30)          null    ,
    R_CLERK_ID       char(30)          null    ,
    ID               int               not null
)
go

/* ============================================================ */
/*   Table: BORROW                                              */
/* ============================================================ */
create table BORROW
(
    READER_ID        char(30)          null    ,
    BOOK_ID          char(30)          null    ,
    BORROW_DATE      datetime          null    ,
    B_CLERK_ID       char(30)          null    ,
    ID               int               not null
```

```
)
go

/* ============================================================ */
/*   Table: FINE_HISTORY                                        */
/* ============================================================ */
create table FINE_HISTORY
(
   READER_ID          char(30)            null    ,
   AMOUNT             money               null    ,
   DAYS               int                 null    ,
   FINE_DATE          datetime            null    ,
   OK_DATE            datetime            null    ,
   ID                 int                 not null
)
go

/* ============================================================ */
/*   Table: FINE                                                */
/* ============================================================ */
create table FINE
(
   READER_ID          char(30)            null    ,
   AMOUNT             money               null    ,
   DAYS               int                 null    ,
   FINE_DATE          datetime            null    ,
   CLERK_ID           char(30)            null    ,
   ID                 int                 not null
)
go

alter table HISTORY
   add constraint FK_HISTORY_REF_52_BOOK foreign key  (BOOK_ID)
      references BOOK (BOOK_ID)
```

```
go

alter table HISTORY
   add constraint FK_HISTORY_REF_58_READER foreign key (READER_ID)
      references READER (READER_ID)
go

alter table BORROW
   add constraint FK_BORROW_REF_49_BOOK foreign key (BOOK_ID)
      references BOOK (BOOK_ID)
go

alter table BORROW
   add constraint FK_BORROW_REF_61_READER foreign key (READER_ID)
      references READER (READER_ID)
go

alter table FINE_HISTORY
   add constraint FK_FINE_HIS_REF_55_READER foreign key
(READER_ID)
      references READER (READER_ID)
go

alter table FINE
   add constraint FK_FINE_REF_64_READER foreign key (READER_ID)
      references READER (READER_ID)
go
```

通过Access的图形界面来建立数据库的方法，本书不作详细介绍，读者可参阅有关Access方面的书。在下面小节中，将具体介绍如何利用Visual C++构建简单的图书管理系统。