

So Chung Yin

6E (19)

ICT 2020SBA

Preface

Library System (Simplified Version)

Miss Chan, the library teacher in ABC College, would like to develop a database system for storing necessary data and facilitating the daily works.

Here belows are some basic information about the library system :

1. Each student can borrow up to 8 books for 14 days.
2. A student can renew the books for another 14 days at most 3 times.
3. There will be a fine of HK\$2 each day for any late return / late renewal.
4. A student can reserve one book and will be informed by email when the book is ready for loan.
5. The books are categorized into different categories, e.g. fiction, science, etc.

As the Database Designer, you are going to provide a solution to this project. As the programs will be developed by other teams, you could focus on the database design and writing necessary SQL statements.

In this project, a prototype of the suggested database schema will be created by MySQL server. Testing mainly focuses on the correctness of field type, primary key constraint, foreign key constraint and DBMS built-in data validation. As the application program is not the scope of this project, user interface design, algorithm design, program coding, program testing & debugging will not be included in this task.

A. Create a prototype

Database with name 2020SBA and the tables with all sample records can be created by the script file create_database.sql

```
-- MySQL Administrator dump 1.4
--
--
-- Server version      5.5.59

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;

/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0
*/;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;

--
-- Create schema 2020sba
--

CREATE DATABASE IF NOT EXISTS 2020sba;
USE 2020sba;

--
-- Definition of table `book`
--

DROP TABLE IF EXISTS `book`;
CREATE TABLE `book` (
  `Bid` char(5) NOT NULL DEFAULT "",
  `title` varchar(40) DEFAULT NULL,
  `cid` char(3) DEFAULT NULL,
  `author` varchar(40) DEFAULT NULL,
  `status` char(1) DEFAULT NULL,
  `reserveby` char(5) DEFAULT NULL,
  PRIMARY KEY (`Bid`),
```

```

    KEY `cid` (`cid`),
    CONSTRAINT `book_ibfk_1` FOREIGN KEY (`cid`) REFERENCES `category` (`Cid`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `book`
--

/*!40000 ALTER TABLE `book` DISABLE KEYS */;
INSERT INTO `book` (`Bid`,`title`,`cid`,`author`,`status`,`reserveby`) VALUES
('b0001','apple','c01','Betty','R','s0001'),
('b0002','orange','c02','Jade','R','s0002'),
('b0003','banana','c03','Sam','S',NULL),
('b0004','berry','c04','Tim','B',NULL),
('b0005','lemon','c05','Kyle','S',NULL);
/*!40000 ALTER TABLE `book` ENABLE KEYS */;

--
-- Definition of trigger `test_on_book`
--

DROP TRIGGER /*!50030 IF EXISTS */ `test_on_book`;

DELIMITER $$

CREATE DEFINER = `root`@`localhost` TRIGGER `test_on_book` BEFORE INSERT ON
`book` FOR EACH ROW begin
    If not((new.bid like ('b_____')) and (mid(new.bid,2,4) between '0001' and '9999'))
then
    Signal sqlstate '45000'
    Set message_text = 'invalid bid value';
end if;
end $$

DELIMITER ;

--
-- Definition of table `category`
--

DROP TABLE IF EXISTS `category`;
CREATE TABLE `category` (
  `Cid` char(3) NOT NULL DEFAULT '',
  `description` varchar(20) DEFAULT NULL,
  PRIMARY KEY (`Cid`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

```

```

--
-- Dumping data for table `category`
--

/*!40000 ALTER TABLE `category` DISABLE KEYS */;
INSERT INTO `category` (`Cid`,`description`) VALUES
('c01','fiction'),
('c02','science fiction'),
('c03','music'),
('c04','history'),
('c05','philosophy');
/*!40000 ALTER TABLE `category` ENABLE KEYS */;

--
-- Definition of trigger `test_on_category`
--

DROP TRIGGER /*!50030 IF EXISTS */ `test_on_category`;

DELIMITER $$

CREATE DEFINER = `root`@`localhost` TRIGGER `test_on_category` BEFORE INSERT
ON `category` FOR EACH ROW begin
    If not((new.cid like ('c__')) and (mid(new.cid,2,4) between '01' and '12')) then
        Signal sqlstate '45000'
        Set message_text = 'invalid cid value';

    end if;
end $$

DELIMITER ;

--
-- Definition of table `loan`
--

DROP TABLE IF EXISTS `loan`;
CREATE TABLE `loan` (
  `Lid` char(5) NOT NULL DEFAULT "",
  `sid` char(5) DEFAULT NULL,
  `bid` char(5) DEFAULT NULL,
  `borrowdate` date DEFAULT NULL,
  `returndate` date DEFAULT NULL,
  `Currentduedate` date DEFAULT NULL,
  `fine` int(11) DEFAULT NULL,
  `renew_date_1` date DEFAULT NULL,
  `renew_date_2` date DEFAULT NULL,

```

```

`renew_date_3` date DEFAULT NULL,
PRIMARY KEY (`Lid`),
KEY `sid` (`sid`),
KEY `bid` (`bid`),
CONSTRAINT `loan_ibfk_1` FOREIGN KEY (`sid`) REFERENCES `student` (`sid`),
CONSTRAINT `loan_ibfk_2` FOREIGN KEY (`bid`) REFERENCES `book` (`Bid`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `loan`
--

/*!40000 ALTER TABLE `loan` DISABLE KEYS */;
INSERT INTO `loan`
(`Lid`,`sid`,`bid`,`borrowdate`,`returndate`,`Currentduedate`,`fine`,`renew_date_1`,`r
enew_date_2`,`renew_date_3`) VALUES
('00001','s0001','b0001','2019-05-01','2019-05-15','2019-05-15',0,NULL,NULL,NULL),
('00002','s0002','b0002','2019-06-01','2019-06-15','2019-06-15',0,NULL,NULL,NULL),
('00003','s0003','b0003','2019-07-01','2019-07-17','2019-07-15',4,NULL,NULL,NULL),

('00004','s0004','b0004','2019-08-01','2019-08-30','2019-08-30',0,'2019-08-10','2019
-08-20',NULL),

('00005','s0005','b0005','2019-09-01','2019-09-20','2019-09-20',2,'2019-09-16',NULL,
NULL);
/*!40000 ALTER TABLE `loan` ENABLE KEYS */;

--
-- Definition of trigger `test_on_loan`
--

DROP TRIGGER /*!50030 IF EXISTS */ `test_on_loan`;

DELIMITER $$

CREATE DEFINER = `root`@`localhost` TRIGGER `test_on_loan` BEFORE INSERT ON
`loan` FOR EACH ROW begin
    If not((new.lid like ('_____')) and (mid(new.lid,1,5) between '00001' and '99999'))
then
        Signal sqlstate '45000'
        Set message_text = 'invalid lid value';
end if;
end $$

DELIMITER ;

--

```

```

-- Definition of table `student`
--

DROP TABLE IF EXISTS `student`;
CREATE TABLE `student` (
  `sid` char(5) NOT NULL DEFAULT "",
  `name` varchar(20) DEFAULT NULL,
  `class` char(2) DEFAULT NULL,
  `classno` int(11) DEFAULT NULL,
  `email` varchar(20) DEFAULT NULL,
  PRIMARY KEY (`sid`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `student`
--

/*!40000 ALTER TABLE `student` DISABLE KEYS */;
INSERT INTO `student` (`sid`,`name`,`class`,`classno`,`email`) VALUES
('s0001','Amy','1A',1,'a@a.com'),
('s0002','Ben','2B',10,'b@b.com'),
('s0003','Chris','3C',15,'c@c.com'),
('s0004','Daisy','4D',20,'d@d.com'),
('s0005','Eric','5E',25,'e@e.com');
/*!40000 ALTER TABLE `student` ENABLE KEYS */;

--
-- Definition of trigger `test_on_student`
--

DROP TRIGGER /*!50030 IF EXISTS */ `test_on_student`;

DELIMITER $$

CREATE DEFINER = `root`@`localhost` TRIGGER `test_on_student` BEFORE INSERT ON
`student` FOR EACH ROW begin
  If not((new.sid like ('s_____')) and (mid(new.sid,2,4) between '0001' and '9999'))
then
  Signal sqlstate '45000'
  Set message_text = 'invalid sid value';
  end if;
end $$

DELIMITER ;

```

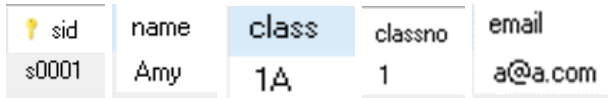
```

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;


```

B. Testing

Test No.1

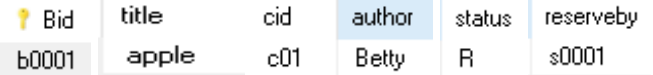
Purpose	See whether a normal record can be successfully inserted into table student
Input	Insert a record with normal test data Insert into student Values('s0001', 'Amy', '1A', 1, 'a@a.com');
Expected output	A new record will be inserted
Actual result	As expected 

Test No.2

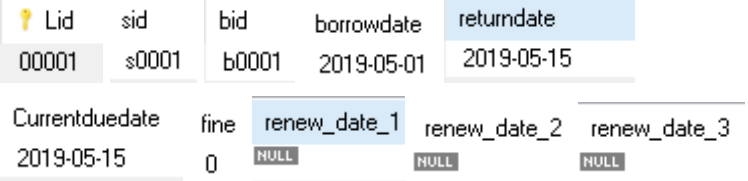
Purpose	See whether a normal record can be successfully inserted into table category
Input	Insert a record with normal test data Insert into category Values('c01', 'fiction');
Expected output	A new record will be inserted
Actual result	As expected 

Test No.3

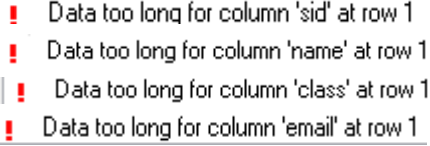
Purpose	See whether a normal record can be successfully inserted into table book
Input	Insert a record with normal test data Insert into book Values('b0001', 'apple', 'c01', 'Betty', 'R', 's0001');

Expected output	A new record will be inserted
Actual result	As expected 

Test No.4



Purpose	See whether a normal record can be successfully inserted into table loan
Input	Insert a record with normal test data Insert into loan Values('00001', 's0001', 'b0001', '2019-05-01', '2019-05-15', '2019-05-15', 0, null, null, null);
Expected output	A new record will be inserted
Actual result	As expected 

Test No.5







Purpose	See whether in table student, field values with length larger than the field width can be detected
Input	Insert a record with sid with 5 characters, name with 21 characters, class with 3 characters, classno and email with 21 characters Insert into student Values('m12345', '123456789012345678901', '123', 1, '123456789012345678901');
Expected output	Proper error message displayed
Actual result	As expected 

Test No.6

Purpose	See whether in table category, field values with length larger than the field width can be detected
Input	Insert a record with cid with 4 characters, description with 21 characters

	Insert into category Values('m012', '123456789012345678901');
Expected output	Proper error message displayed
Actual result	As expected  Data too long for column 'Cid' at row 1  Data too long for column 'description' at row 1

Test No.7

Purpose	See whether in table book, field values with length larger than the field width can be detected
Input	Insert a record with bid with 5 characters, title with 41 characters, cid with 4 characters, author with 41 characters, status with 2 characters, sid with 6 characters Insert into book Values('m12345', '12345678901234567890123456789012345678901', 'm123', '12345678901234567890123456789012345678901', '12', 'm12345');
Expected output	Proper error message displayed
Actual result	As expected  Data too long for column 'Bid' at row 1  Data too long for column 'title' at row 1  Data too long for column 'cid' at row 1  Data too long for column 'author' at row 1  Data too long for column 'status' at row 1  Data too long for column 'reserveby' at row 1

Test No.8

Purpose	See whether in table loan, field values with length larger than the field width can be detected
Input	Insert a record with lid with 6 characters, sid with 6 characters, bid with 6 characters, while borrowdate, returndate, currentduedate, fine, renew_date_1, renew_date_2, renew_date_3 all with normal test data Insert into loan Values('m12345', 'm12345', 'm12345', '2019-05-01', '2019-05-15', '2019-05-15', 0, null, null, null);
Expected output	Proper error message displayed
Actual result	As expected

	<p>❗ Data too long for column 'Lid' at row 1</p> <p>❗ Data too long for column 'sid' at row 1</p> <p>❗ Data too long for column 'bid' at row 1</p>
--	----------------------------------------------------------------------------------------------------------------------------------------------------

Test No.9

Purpose	See whether in table student, numeric field with character can be detected
Input	<p>Insert a record with sid, name, class, email with normal test data but classno with a character</p> <p>Insert into student Values(('s0002','Ben','2B','s','b@b.com');</p>
Expected output	Proper error message displayed
Actual result	<p>As expected</p> <p>❗ Incorrect integer value: 's' for column 'classno' at row 1</p>

Test No.10

Purpose	See whether in table loan, numeric field and date field either with character can be detected
Input	<p>Insert a record with lid, sid, bid with normal test data but borrowdate, returndate, currentduedate, fine, renew_date_1, renew_date_2, renew_date_3 with a character</p> <p>Insert into loan Values('00002','s0002','b0002','s','s','s','s','s','s','s');</p>
Expected output	Proper error message displayed
Actual result	<p>As expected</p> <p>❗ Incorrect date value: 's' for column 'borrowdate' at row 1</p> <p>❗ Incorrect date value: 's' for column 'returndate' at row 1</p> <p>❗ Incorrect date value: 's' for column 'Currentduedate' at row 1</p> <p>❗ Incorrect integer value: 's' for column 'fine' at row 1</p> <p>❗ Incorrect date value: 's' for column 'renew_date_1' at row 1</p> <p>❗ Incorrect date value: 's' for column 'renew_date_2' at row 1</p> <p>❗ Incorrect date value: 's' for column 'renew_date_3' at row 1</p>

Test No.11

Purpose	See whether the primary key constraint in table student is implemented properly
Input	<p>Insert a record with existing sid</p> <p>Insert into student Values('s0001','Ben','2B',10,'b@b.com');</p>
Expected output	Proper error message displayed

Actual result	As expected ! Duplicate entry 's0001' for key 'PRIMARY'
---------------	------------------------------------------------------------

Test No.12

Purpose	See whether the primary key constraint in table category is implemented properly
Input	Insert a record with existing cid Insert into category Values('c01', 'science fiction');
Expected output	Proper error message displayed
Actual result	As expected ! Duplicate entry 'c01' for key 'PRIMARY'

Test No.13

Purpose	See whether the primary key constraint in table book is implemented properly
Input	Insert a record with existing bid Insert into book Values('b0001', 'orange', 'c02', 'Jade', 'R', 's0002');
Expected output	Proper error message displayed
Actual result	As expected ! Duplicate entry 'b0001' for key 'PRIMARY'

Test No.14

Purpose	See whether the primary key constraint in table loan is implemented properly
Input	Insert a record with existing lid Insert into loan Values('00001', 's0002', 'b0002', '2019-06-01', '2019-06-15', '2019-06-15', 0, null, null, null);
Expected output	Proper error message displayed
Actual result	As expected ! Duplicate entry '00001' for key 'PRIMARY'

Test No.15

Purpose	See whether the foreign key constraint in table book is implemented properly
Input	Insert a record with non-existing bid, title, cid, author and sid

	Insert into book Values('b1234', '1234567890123456789012345678901234567890', 'c12', '1234567890123456789012345678901234567890', '1', 's1234')
Expected output	Proper error message displayed
Actual result	As expected ❗ Cannot add or update a child row: a foreign key constraint fails ('library`.`book`, CONSTRAINT `book_ibfk_1` FOREIGN KEY (`cid`) REFERENCES `category` (`Cid`))

Test No.16


Purpose	See whether the foreign key constraint in table loan is implemented properly
Input	Insert a record with non-existing lid, sid, bid, but borrowdate, returndate, currentduedate, fine, renew_date_1, renew_date_2, renew_date_3 with normal test data Insert into loan Values('12345', 's1234', 'b1234', '2019-05-01', '2019-05-15', '2019-05-15',0,null,null,null)
Expected output	Proper error message displayed
Actual result	As expected ❗ Cannot add or update a child row: a foreign key constraint fails ('library`.`loan`, CONSTRAINT `loan_ibfk_1` FOREIGN KEY (`sid`) REFERENCES `student` (`sid`) ❗ Cannot add or update a child row: a foreign key constraint fails ('library`.`loan`, CONSTRAINT `loan_ibfk_2` FOREIGN KEY (`bid`) REFERENCES `book` (`Bid`))

Test No.17


Purpose	See whether invalid classno value can be detected
Input	Insert a record with invalid classno 0 Insert into student Values('s0002', 'Ben', '2B', 0, 'b@b.com');
Expected output	Proper error message displayed
Actual result	As expected ❗ invalid classno value
Related statement	DELIMITER \$\$ CREATE DEFINER = `root`@`localhost` TRIGGER `test_on_student` BEFORE INSERT ON `student` FOR EACH ROW begin If (new.classno<1) or (new.classno>40) then Signal sqlstate '45000'

	Set message_text = 'invalid classno value'; End if; end \$\$ DELIMITER ;
--	---------------------------------------------------------------------------------------

Test No.18

Purpose	See whether invalid sid value can be detected
Input	Insert a record with invalid sid value 'm0001'. Insert into student Values('m0001', 'Ben', '2B', 1, 'b@b.com');
Expected output	Proper error message displayed
Actual result	As expected  invalid sid value
Related statement	DELIMITER \$\$ CREATE DEFINER = `root`@`localhost` TRIGGER `test_on_student` BEFORE INSERT ON `student` FOR EACH ROW begin If not((new.sid like ('s_____')) and (mid(new.sid,2,4) between '0001' and '9999')) then Signal sqlstate '45000' Set message_text = 'invalid sid value'; end if; end \$\$ DELIMITER ;

Test No.19

Purpose	See whether invalid cid value can be detected
Input	Insert a record with invalid cid value 'c13' Insert into category Values('c13', 'science fiction');
Expected output	Proper error message displayed
Actual result	As expected  invalid cid value
Related statement	DELIMITER \$\$ CREATE DEFINER = `root`@`localhost` TRIGGER `test_on_category` BEFORE INSERT ON `category` FOR EACH ROW begin

	<pre> If not((new.cid like ('c__')) and (mid(new.cid,2,4) between '01' and '12')) then Signal sqlstate '45000' Set message_text = 'invalid cid value'; end if; end \$\$ DELIMITER ; </pre>
--	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Test No.20

Purpose	See whether invalid lid value can be detected
Input	<p>Insert a record with invalid lid value '00000'</p> <p>Insert into loan Values('00000','s0002','b0002','2019-06-01','2019-06-15','2019-06-15',0,NULL,NULL,NULL);</p>
Expected output	Proper error message displayed
Actual result	As expected
Related statement	<pre> DELIMITER \$\$ CREATE DEFINER = `root`@`localhost` TRIGGER `test_on_loan` BEFORE INSERT ON `loan` FOR EACH ROW begin If not((new.lid like ('_____')) and (mid(new.lid,1,5) between '00001' and '99999')) then Signal sqlstate '45000' Set message_text = 'invalid lid value'; end if; end \$\$ DELIMITER ; </pre>

Test No.21

Purpose	See whether invalid bid value can be detected
Input	<p>Insert a record with invalid bid value 'm0002'</p> <p>Insert into book Values('m0002','orange','c02','Jade','R','s0002')</p>
Expected output	Proper error message displayed
Actual result	As expected
Related statement	DELIMITER \$\$

	<pre> CREATE DEFINER = `root`@`localhost` TRIGGER `test_on_book` BEFORE INSERT ON `book` FOR EACH ROW begin If not((new.bid like ('b_____')) and (mid(new.bid,2,4) between '0001' and '9999')) then Signal sqlstate '45000' Set message_text = 'invalid bid value'; end if; end \$\$ DELIMITER ; </pre>
--	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Test No.22

Purpose	See whether invalid status value can be detected
Input	<p>Insert a record with invalid status 'A'</p> <p>Insert into book Values('b0002','orange','c02','Jade','A',null);</p>
Expected output	Proper error message displayed
Actual result	As expected
Related statement	<pre> DELIMITER \$\$ CREATE DEFINER = `root`@`localhost` TRIGGER `test_on_book` BEFORE INSERT ON `book` FOR EACH ROW begin If (new.status) <> 'S' and (new.status) <>'R' and (new.status) <> 'B' then Signal sqlstate '45000' Set message_text = 'invalid STATUS'; end if; end \$\$ DELIMITER ; </pre>

C. Evaluation

Discussion: Field 'reservedby'

For this field, it cannot deal with the reservation efficiently. Having this field can help users to check for students if they have reserved a book by checking his or her presence or absence of their sid in the book records. However there is not always a student who wants to reserve books. Therefore, the 'reserved

by' fields will result in many null values among the book records, occupying a part of storage space.

Also, the 'reserved by' field doesn't have any due date after reserving it. If one does not take the reserved book after receiving the email from library, not only will the reserved book not be borrowed, the field will also be unable to update in a short period of time, or it may require others to update it manually, thus is not convenient for users to use the database. And this field cannot fix the reservation problems easily.

Discussion: table 'loan'

For this table loan, it has its advantages and disadvantages. Table loan contains fields of current due dates ,renew dates and return dates. This can help calculate the fine of each loan record easily, increasing the efficiency to sum up the fine for every student. Also, the renewal dates can help check how many times the student has done the renewal for the book, which gives users a better understanding of the loan records. It is good in terms of user friendliness.

However, not every student would renew a book, let alone the possibility of renewing it a third time. Even if they renew once, there are two renew date fields to be left null. Therefore, it causes a higher chance to have many null values in renew dates.

Any new field to be added

The due date of reservation should be added. Students may sometimes forget or become unwilling to take the reserved book for some reasons. If the book record still does not have a due date, it causes many troubles in the reserving and borrowing process. For instance, others may want to borrow that book and he or she is waiting for the former borrower to return it. But, the book is being held as the book is still in reserved status and waits for the one who reserved the book to take it. As a result, we need to add the due date of reservation so that it can help users to know by what time to take the reserved book, and the database can be updated more easily, since it can just change the book status back to on the shelf after the due date.

Necessity of derived fields

As we all know, the fine of loan records can be derived from the existing field. We just need to compare the renewal date or return date to the current due date, and count for the late days and times 2. Then the fine is calculated. The calculations seem to be so easy that they can substitute the presence of the field. However, we need to know that students can borrow at most eight books every time. If one is late for all the eight borrowed books, the calculations

would be more complicated and not efficient way. Having this field allows users to add up all of them according to the sid, which can easily get the same result like the above one. In order to have better performance and be more user-friendly, this field is a wise choice to remain in the table.

Pros and cons of the database design

This database is designed with 4 tables. The separation of table book and table category can reduce the redundancy of the names of the categories. The status field of a book is a good design that can tell users the condition of the book while taking a character only, which proves it to be a wise design while thinking of user friendliness.

Also, there are some fields not in the table but can be calculated from the existing fields. For example, the upper limit of times to borrow books can be calculated by counting the loan records containing the sid of students and checking if renew date fields are null respectively. Then it truly can reduce the space to store this extra field. And the maximum number of renewing one book can also be found as it just checks how many renew date fields are not null. These easy calculations are good to substitute more fields to store those information.

However, the database has some limitations. It will be full easily for table loan. If one student borrows eight books every time, he will create eight records for each book. Then, after keeping Borrow 8 books per two weeks, it is not difficult to use up all the loan records allowed in a short period of time. As a consequence, we can see that the number of loan record ID is not enough.

Also, there is no punishment for late taking of reserved books. If students do not realise the situation, there would only be more and more late taking of reserved books. Thus, it would become hard to control the normal flow of the library system, which is no good to the library.

Concept of relational database

For this database, it really has a very well-designed and correct relationships between tables and tables. From them, we can easily understand how it works. Each student may have one or more loan records while each loan record must always belong to one and only one student. Students may want to borrow some books to read, which will make a record of each book for him or her. Each book may have one or more loan records while each loan record must only contain one and only one book. There is always a book being borrowed then a loan record will be made for it. However, we know that one book could have more than one category, so we can find that each book may have one or many categories while each category must have one or many books. Therefore, there is a multi-value attribute cid in table book which is not

separated with the table book. It still is in the unnormalized form(UNF). There is a necessity to create a book-category table to store the categories of each book in order to make it in third normalized form(3NF).

Major change

Since this database is not very perfect, here is a suggestion to alter the database. Any adding of fields about reservations are not enough to cope with the problem. As a result, I would suggest making a new table of reservation. It contains the rid, sid, bid, and due date. When a student reserves a book, it just needs to build a new record including certain sid and bid. Students would be informed by email that he or she can take the book after searching the email address from the sid. And the book would be changed to R in status.

This change is a must to be carried out. First, the original one results in many null values. Not every book will be reserved at anytime. Then, if no one reserve the book, it gets a null value in the record. So, it may consume much space. Using this table can not only be unnecessary to design any field for it, it can also give a clear information of reservation to the user, allowing them to manage those reserve records more efficiently, and easy to check for due date of taking the reserved books. These advantages can be brought with just one more table with the necessary information. So it can help a lot in the future.