数据库系统及应用实验报告-Lab01

> 姓名： 张劲暾
>
> 学号： PB16111485
>
> 实验： SQL & PL/SQL
>
> 实验环境：
>
> - DBMS: Oracle18.3
> - IDE: PL/SQL Developer Version 13.0.3.1902 (64 bit)

实验内容

设某图书馆数据库包含下面的基本表：

- **Book(ID: char(8), name:varchar2(10), author:varchar2(10), price:float, status: int)**
  1. 图书号(ID)为主键，书名不能为空。
  2. 状态(status)为 1 表示书被借出，0 表示在馆，默认值为 0。
- **Reader(ID:char(8), name:varchar2(10), age:int, address:varchar2(20))**
  1. 读者号 ID 为主键。
- **Borrow(book_ID:char(8), Reader_ID:char(8), Brrrow_Date:date, Return_Date:date)**
  1. 还期Return_Date为NULL表示该书未还
  2. 主键为（图书号，读者号）
  3. 图书号为外键，引用图书表的图书号
  4. 读者号为外键，引用读者表的读者号

实验步骤与结果

Step 1：创建上述基本表，并插入部分测试数据

创建基本表：

``` plsql
/*=== 清除原有表格对象 ===*/
Drop Table Book     Cascade Constraints;
Drop Table Reader   Cascade Constraints;
Drop Table Borrow   Cascade Constraints;
/*=== 创建基本表 ===*/
/*=============================================================================================*/
create table Book(
    ID      char(8)     Constraint Book_PK Primary Key,    -- 图书号
    name    varchar2(8) NOT NULL ,                         -- 书名
    author  varchar2(8),
    price   float,
    /* 状态，status = 1 表示书被借出，status = 0 表示在馆，默认值为0 */
    status  int DEFAULT 0,
```

```
    ------------------------------------------------------------------
    Constraint status_Choise    Check (status = 1 or status = 0),
    Constraint price_floor      Check (price >= 0)
);
/*=============================================================================================*
/
create table Reader(
    ID       char(8)     Constraint Reader_PK Primary Key,  --读者号
    name     varchar2(10),
    age      int,
    address  varchar2(20),
    ------------------------------------------------------------------
    Constraint age_floor    Check (age > 0)
);
/*=============================================================================================*
/
create table Borrow(
    book_ID     char(8),
    Reader_ID   char(8),
    Borrow_Date date,
    Return_Date date,
    Constraint FK_book_ID   Foreign Key(book_ID)    References Book(ID),
    Constraint FK_Reader_ID Foreign Key(Reader_ID)  References Reader(ID),
    Constraint Borrow_PK    Primary Key(book_ID,Reader_ID)
);
```

加入测试数据

```plsql
/*=============================================================================================*
/
/*=== 插入数据简单测试 ===*/
/*=== 避免了执行多个SQL语句多次连接数据库的开销 ====*/
insert all
-------------------------------------------------------------------------------------------------
--------
into    Book(ID, name, author, price, status)   values
('0001','database','C.J.Date',23.33,0)
into    Book(ID, name, author, price, status)   values
('0002','datastru','mjh',6.66,1)
into    Book(ID, name, author, price, status)   values
('0003','algorith','gnj',99.99,0)
-------------------------------------------------------------------------------------------------
--------
into    Reader(ID, name, age, address)  values  ('PB1485','zjt',18,'x4-414')
into    Reader(ID, name, age, address)  values  ('PB1666','dalao1',28,'d333-222')
into    Reader(ID, name, age, address)  values  ('PB4554','dalao2',44,'x5-233')
-------------------------------------------------------------------------------------------------
--------
into    Borrow(book_ID, Reader_ID, Borrow_Date, Return_Date)    values
('0002','PB1485',to_date('04/13/2019','mm/dd/yyyy'),NULL)
-------------------------------------------------------------------------------------------------
--------
select 1 from DUAL;
/*=============================================================================================*
/
```

简单测试

```plsql
/*=============================================================================================*/
select Book.Name, Reader.Name,Borrow.Borrow_Date
from Book, Reader, Borrow
where
Book.Id = Borrow.Book_Id and
Reader.Id = Borrow.Reader_Id;
/*=============================================================================================*/
```

结果：

```xml
                NAME     NAME     BORROW_DATE
1      1    datastru     zjt      2019/4/13
```

Step 2：设计例子，验证实体完整性、参照完整性、用户自定义完整性

实体完整性检查

```
insert into Book(ID, name, author, price, status) values (NULL,'database','C.J.Date',23.33,0);
```

报错：ORA-01400: 无法将 NULL 插入 ("SYSTEM"."BOOK"."ID")

```
insert into Reader(ID, name, age, address) values (NULL,'lala',88, 'x3-333');
```

报错：ORA-01400: 无法将 NULL 插入 ("SYSTEM"."READER"."ID")

```
insert into Borrow(book_ID, Reader_ID, Borrow_Date, Return_Date) values
(NULL,'PB1485',to_date('04/13/2019','mm/dd/yyyy'),NULL);
```

报错：ORA-01400: 无法将 NULL 插入 ("SYSTEM"."BORROW"."BOOK_ID")

```
insert into Borrow(book_ID, Reader_ID, Borrow_Date, Return_Date) values
('0002',NULL,to_date('04/13/2019','mm/dd/yyyy'),NULL);
```

报错：ORA-01400: 无法将 NULL 插入 ("SYSTEM"."BORROW"."READER_ID")

参照完整性检查

```
insert into Borrow(book_ID, Reader_ID, Borrow_Date, Return_Date) values
('0005','PB1485',to_date('04/13/2019','mm/dd/yyyy'),NULL);
```

报错：ORA-02291: 违反完整约束条件 (SYSTEM.FK_BOOK_ID) - 未找到父项关键字

```
insert into Borrow(book_ID, Reader_ID, Borrow_Date, Return_Date) values
('0002','PB1488',to_date('04/13/2019','mm/dd/yyyy'),NULL);
```

报错：ORA-02291: 违反完整约束条件 (SYSTEM.FK_READER_ID) - 未找到父项关键字

用户自定义完整性检查

```
insert into Book(ID, name, author, price, status) values ('0004','architect','abc',23.33,7);
```

报错：ORA-12899: 列 "SYSTEM"."BOOK"."NAME" 的值太大 (实际值: 9, 最大值: 8)

```
insert into Book(ID, name, author, price, status) values ('0004','architct','abc',-23.33,7);
```

报错：ORA-02290: 违反检查约束条件 (SYSTEM.PRICE_FLOOR)

```
insert into Reader(ID, name, age, address) values ('PB5656','lala',-9, 'x3-333');
```

报错：ORA-12899: 列 "SYSTEM"."BOOK"."NAME" 的值太大 (实际值: 9, 最大值: 8)

Step 3：SQL语言完成下面小题，并测试运行结果

辅助数据：

```plsql
insert all
into    Book(ID, name, author, price, status)   values
('0004','Oracleaa','Ullman',23.33,0)
into    Book(ID, name, author, price, status)   values
('0005','Oraclebb','Ullman',6.66,1)
into    Book(ID, name, author, price, status)   values
('0006','cccccc','Ullman',99.99,0)
into    Book(ID, name, author, price, status)   values
('0007','dddddd','Ullman',23.33,0)
into    Book(ID, name, author, price, status)   values  ('0008','eeeeee','adam',6.66,1)
into    Book(ID, name, author, price, status)   values
('0009','Oracleff','ustc',99.99,0)
into    Book(ID, name, author, price, status)   values
('0010','gggggg','stanford',23.33,0)
into    Book(ID, name, author, price, status)   values
('0011','Oraclehh','cmu',6.66,1)
into    Book(ID, name, author, price, status)   values
('0012','Oracleii','ucb',99.99,0)
-------------------------------------------------------------------------------------------------
--------
into    Reader(ID, name, age, address)  values  ('PB0001','Rose',18,'x4-414')
into    Reader(ID, name, age, address)  values  ('PB0002','李林',28,'d333-222')
-------------------------------------------------------------------------------------------------
--------
```

```
into    Borrow(book_ID, Reader_ID, Borrow_Date, Return_Date)
values
('0002','PB0001',to_date('04/13/2018','mm/dd/yyyy'),to_date('05/13/2018','mm/dd/yyyy'))
into    Borrow(book_ID, Reader_ID, Borrow_Date, Return_Date)
values
('0005','PB0001',to_date('04/16/2018','mm/dd/yyyy'),to_date('06/13/2018','mm/dd/yyyy'))
into    Borrow(book_ID, Reader_ID, Borrow_Date, Return_Date)
values
('0007','PB0001',to_date('11/16/2018','mm/dd/yyyy'),to_date('01/13/2019','mm/dd/yyyy'))
into    Borrow(book_ID, Reader_ID, Borrow_Date, Return_Date)
values  ('0011','PB0001',to_date('02/13/2017','mm/dd/yyyy'),NULL)
into    Borrow(book_ID, Reader_ID, Borrow_Date, Return_Date)
values
('0003','PB0002',to_date('04/13/2018','mm/dd/yyyy'),to_date('05/13/2018','mm/dd/yyyy'))
into    Borrow(book_ID, Reader_ID, Borrow_Date, Return_Date)
values
('0006','PB0002',to_date('04/16/2018','mm/dd/yyyy'),to_date('06/13/2018','mm/dd/yyyy'))
into    Borrow(book_ID, Reader_ID, Borrow_Date, Return_Date)
values  ('0010','PB0002',to_date('02/13/2019','mm/dd/yyyy'),NULL)
-------------------------------------------------------------------------------------------------
--------
select 1 from DUAL;
```

检索读者Rose 的读者号和地址

```plsql
select ID, address
from Reader
where name = 'Rose';
```

结果：

```xml
             ID      ADDRESS
1     PB0001      x4-414
```

检索读者Rose 所借阅读书（包括已还和未还图书）的图书名和借期

```plsql
select Book.Name, Borrow.Borrow_Date
from Book, Reader, Borrow
where
    Reader.Name = 'Rose' and
    Reader.Id = Borrow.Reader_Id and
    Book.Id = Borrow.Book_Id;
```

结果：

```xml
             NAME    BORROW_DATE
1    datastru    2018/4/13
2    Oraclebb    2018/4/16
3    dddddd      2018/11/16
4    Oraclehh    2017/2/13
```

检索未借阅图书的读者姓名

```plsql
select name
from Reader
where ID NOT IN
(
    select distinct ID
    from Reader
);
```

结果：

```xml
         NAME
    1    dalao1
    2    dalao2
```

检索Ullman 所写的书的书名和单价

```plsql
select name, price
from Book
where author = 'Ullman';
```

结果：

```xml
         NAME      PRICE
    1    Oracleaa    23.33
    2    Oraclebb    6.66
    3    cccccc      99.99
    4    dddddd      23.33
```

检索读者“李林”借阅未还的图书的图书号和书名

```plsql
select Book.ID, Book.name
from Book, Reader, Borrow
where
    Reader.name = '李林' and
    Reader.Id = Borrow.Reader_Id and
    Book.ID = Borrow.Book_Id and
    Borrow.Return_Date is NUll
;
```

结果：

```xml
         ID          NAME
    1    0010        gggggg
```

检索借阅图书数目超过 3 本的读者姓名

```plsql
select Reader.Name
from Reader,    (
                    select count(Book_Id) as borrow_count, Reader_Id
                    from Borrow
                    Group by Reader_Id
                )BorrowCount
where
    BorrowCount.borrow_count > 3 and
    BorrowCount.Reader_Id = Reader.Id
;
```

结果：

```xml
           NAME
    1      Rose
```

检索没有借阅读者“李林”所借的任何一本书的读者姓名和读者号

```plsql
select distinct Reader.Id, Reader.Name
from Reader, Borrow
where
    Reader.Id = Borrow.Reader_Id and
    Borrow.Book_Id NOT IN (
        select Borrow.Book_Id
        from Reader, Borrow
        where
            Reader.Name = '李林' and
            Reader.Id = Borrow.Reader_Id
    )
;
```

结果：

```xml
              ID      NAME
    1    PB1485      zjt
    2    PB0001      Rose
```

检索书名中包含“Oracle”的图书书名及图书号

```plsql
select Id, name
from Book
where name Like '%Oracle%';
```

结果：

```xml
             ID          NAME

         1   0004        Oracleaa
         2   0005        Oraclebb
         3   0009        Oracleff
         4   0011        Oraclehh
         5   0012        Oracleii
```

创建一个读者借书信息的视图，该视图包含读者号、姓名、所借图书号、图书名 和借期；并使用该视图查询最近一年所有读者的读者号以及所借阅的不同图书数

```plsql
Drop View BorrowInfo;
Create View BorrowInfo (Reader_Id, Reader_Name, Book_Id, Book_Name, Borrow_Date)
AS  Select Borrow.Reader_Id, Reader.Name, Borrow.Book_Id, Book.Name, Borrow.Borrow_Date
    From Book, Reader, Borrow
    Where
        Reader.Id = Borrow.Reader_Id and
        Book.Id   = Borrow.Book_Id
With Read Only;
------------------------
select *
from BorrowInfo
where
    -- ROUND(TO_NUMBER(BorrowInfo.Borrow_Date - SYSDATE())) <= 365
    Borrow_Date >= sysdate - 365;
;
```

结果：

```xml
         READER_ID   READER_NAME BOOK_ID     BOOK_NAME   BORROW_DATE
1        PB1485      zjt          0002       datastru    2019/4/13
2        PB0001      Rose         0007       dddddd      2018/11/16
3        PB0002      李林         0010         ggggggg     2019/2/13
```

Step 4：设计存储过程，实现对 Book 表的 ID的修改

```plsql
Create or Replace Procedure Change_Book_ID(
    oldBookId IN char,
    newBookId IN char
)
AS
    tempCount   number;
    oldNameNotFound Exception;
    newNameOccupied Exception;
BEGIN
    ------------------------------------------------------------------------------
    SELECT COUNT(*) INTO tempCount FROM DUAL WHERE EXISTS(SELECT NULL FROM Book WHERE
ID = newBookId);
    IF (tempCount = 1) Then
        raise newNameOccupied;
```

```
    ELSE
        SELECT COUNT(*) INTO tempCount FROM DUAL WHERE EXISTS(SELECT NULL FROM Book WHERE ID = oldBookId);
        IF (tempCount = 0) THEN
            raise oldNameNotFound;
        ELSE
            execute immediate 'Alter Table Borrow Drop Constraint FK_book_ID';
            ---------------------------------------
            Update Book
            Set ID = newBookId
            Where ID = oldBookId;
            ---------------------------------------
            Update Borrow
            Set book_Id = newBookId
            Where book_Id = oldBookId;
            ---------------------------------------
            execute immediate 'Alter Table Borrow Add Constraint FK_book_ID Foreign Key(book_ID) References Book(ID)';
        End IF;
    End IF;
    ------------------------------------------------------------------------------------------
    EXCEPTION
        When oldNameNotFound Then
            raise_application_error(-20001,'需要修改的记录不存在');
        When newNameOccupied Then
            raise_application_error(-20002,'命名冲突');
        When Others Then
            DBMS_OUTPUT.PUT_LINE('错误号: '||SQLCODE||'    错误描述: '||SQLERRM);
    ------------------------------------------------------------------------------------------
END Change_Book_ID;
/
```

结果:

```shell
SQL> select * from Book;

ID               NAME            AUTHOR               PRICE     STATUS
---------------- --------------- ---------------- ---------- ----------
0001             database        C.J.Date              23.33          0
0002             datastru        mjh                    6.66          1
0003             algorith        gnj                   99.99          0
0004             Oracleaa        Ullman                23.33          0
0005             Oraclebb        Ullman                 6.66          1
0006             cccccc          Ullman                99.99          0
0007             dddddd          Ullman                23.33          0
0008             eeeeee          adam                   6.66          1
0009             Oracleff        ustc                  99.99          0
0010             gggggg          stanford              23.33          0
0011             Oraclehh        cmu                    6.66          1

ID               NAME            AUTHOR               PRICE     STATUS
---------------- --------------- ---------------- ---------- ----------
0012             Oracleii        ucb                   99.99          0

已选择 12 行。

SQL> select * from Borrow;

```

```
BOOK_ID          READER_ID        BORROW_DATE    RETURN_DATE
---------------- ---------------- -------------- --------------
0002             PB1485           13-4月 -19
0002             PB0001           13-4月 -18     13-5月 -18
0005             PB0001           16-4月 -18     13-6月 -18
0007             PB0001           16-11月-18     13-1月 -19
0011             PB0001           13-2月 -17
0003             PB0002           13-4月 -18     13-5月 -18
0006             PB0002           16-4月 -18     13-6月 -18
0010             PB0002           13-2月 -19
0012             PB1485           14-4月 -19     14-5月 -19

已选择 9 行。

SQL> exec Change_Book_ID('0002','0020');

PL/SQL 过程已成功完成。

SQL> select * from Book;

ID               NAME             AUTHOR               PRICE     STATUS
---------------- ---------------- ---------------- ---------- ----------
0001             database         C.J.Date              23.33          0
0020             datastru         mjh                    6.66          0
0003             algorith         gnj                   99.99          0
0004             Oracleaa         Ullman                23.33          0
0005             Oraclebb         Ullman                 6.66          1
0006             cccccc           Ullman                99.99          0
0007             dddddd           Ullman                23.33          0
0008             eeeeee           adam                   6.66          1
0009             Oracleff         ustc                  99.99          0
0010             gggggg           stanford              23.33          0
0011             Oraclehh         cmu                    6.66          1

ID               NAME             AUTHOR               PRICE     STATUS
---------------- ---------------- ---------------- ---------- ----------
0012             Oracleii         ucb                   99.99          0

已选择 12 行。

SQL> select * from Borrow;

BOOK_ID          READER_ID        BORROW_DATE    RETURN_DATE
---------------- ---------------- -------------- --------------
0020             PB1485           13-4月 -19
0020             PB0001           13-4月 -18     13-5月 -18
0005             PB0001           16-4月 -18     13-6月 -18
0007             PB0001           16-11月-18     13-1月 -19
0011             PB0001           13-2月 -17
0003             PB0002           13-4月 -18     13-5月 -18
0006             PB0002           16-4月 -18     13-6月 -18
0010             PB0002           13-2月 -19
0012             PB1485           14-4月 -19     14-5月 -19

已选择 9 行。

SQL>
```

Step 5: 设计触发器

> 实现：
>
> - 当一本书被借出时，自动将 Book 表中相应图书的 status 修改为 1；
> - 当某本书被归还时，自动将 Book 表中相应图书的 status 修改为 0。

```plsql
Create or Replace Trigger BookStatusChange
----------------------------------------------------------------------------------------
After Insert Or Update On Borrow
----------------------------------------------------------------------------------------
For Each Row
Begin
    ------------------------------------------------------------------------------------
    if :old.Book_Id is NULL then
        Update Book Set status = 1 where ID = :new.Book_Id;
    else
        Update Book Set status = 0 where ID = :new.Book_Id;
    end if;
    ------------------------------------------------------------------------------------
End;
/
-- test --
select * from Book;
insert into Borrow (book_ID, Reader_ID, Borrow_Date, Return_Date)
values ('0012', 'PB1485', to_date('04/14/2019','mm/dd/yyyy'), NULL);
select * from Book;
-----
update Borrow
   set Return_Date = to_date('05/14/2019','mm/dd/yyyy')
 where book_ID = '0012';
select * from Book;
```

简单测试：

```plsql
-- test --
select * from Book;
insert into Borrow (book_ID, Reader_ID, Borrow_Date, Return_Date)
values ('0012', 'PB1485', to_date('04/14/2019','mm/dd/yyyy'), NULL);
select * from Book;
-----
update Borrow
   set Return_Date = to_date('05/14/2019','mm/dd/yyyy')
 where book_ID = '0012';
select * from Book;
```

结果：

第一次`select * from Book;`

```xml
      ID          NAME        AUTHOR      PRICE   STATUS
1     0001        database    C.J.Date    23.33   0
2     0002        datastru    mjh         6.66    1
3     0003        algorith    gnj         99.99   0
4     0004        Oracleaa    Ullman      23.33   0
5     0005        Oraclebb    Ullman      6.66    1
```

```
  7  6   0006        cccccc      Ullman      99.99   0
  8  7   0007        dddddd      Ullman      23.33   0
  9  8   0008        eeeeee      adam        6.66    1
 10  9   0009        Oracleff    ustc        99.99   0
 11  10  0010        gggggg      stanford    23.33   0
 12  11  0011        Oraclehh    cmu         6.66    1
 13  12  0012        Oracleii    ucb         99.99   0
 14
```

第二次`select * from Book;`

```xml
      ID          NAME        AUTHOR      PRICE   STATUS
1     0001        database    C.J.Date    23.33   0
2     0002        datastru    mjh         6.66    1
3     0003        algorith    gnj         99.99   0
4     0004        Oracleaa    Ullman      23.33   0
5     0005        Oraclebb    Ullman      6.66    1
6     0006        cccccc      Ullman      99.99   0
7     0007        dddddd      Ullman      23.33   0
8     0008        eeeeee      adam        6.66    1
9     0009        Oracleff    ustc        99.99   0
10    0010        gggggg      stanford    23.33   0
11    0011        Oraclehh    cmu         6.66    1
12    0012        Oracleii    ucb         99.99   1
13
14
```

第三次`select * from Book;`

```xml
      ID          NAME        AUTHOR      PRICE   STATUS
1     0001        database    C.J.Date    23.33   0
2     0002        datastru    mjh         6.66    1
3     0003        algorith    gnj         99.99   0
4     0004        Oracleaa    Ullman      23.33   0
5     0005        Oraclebb    Ullman      6.66    1
6     0006        cccccc      Ullman      99.99   0
7     0007        dddddd      Ullman      23.33   0
8     0008        eeeeee      adam        6.66    1
9     0009        Oracleff    ustc        99.99   0
10    0010        gggggg      stanford    23.33   0
11    0011        Oraclehh    cmu         6.66    1
12    0012        Oracleii    ucb         99.99   0
13
14
```