**ASSIGNMENT**

Student name: DANG VIET ANH | HE151359

Student name: TRAN XUAN QUANG | HE153654

Student name: DOAN HAI PHONG | HE151364

Teacher: nguyen THI BICH THUY

DBI202 – DATABASE OF library MANAGEMENT SYSTEM

November 9, 2021

# TABLE OF CONTENTS

## INTRODUCE THE PROBLEM…………………………………………………………………………………………….

### Describe the problem……………………………………………………………………………………………………….

### Goals…………………………………………………………………………………………………………………………………..

## entity – relationship – er…………………………………………………………………………………………

### SET-UP ENTITY – RELATIONSHIP …………………………………………………………………………………………..

### difinITION entity – attributE ………………………………………………………………………………………….

## Database and ENTITY RELATIONSHIP DIAGRAM (ERD)………………………………………………..

### PUBLISHER……………………………………………………………………………………………………………………………

### KINDOFBOOK……………………………………………………………………………………………………………………….

### BOOKS….………………………………………………………………………………………………………………………………

### STAFFS…………………………………………………………………………………………………………………………………..

### READERS……………………………………………………………………………………………………………………………….

### BORROW……..……………………………………………………………………………………………………………………….

### BORROWBOOKS\_DETAILS……..………………………………………………………………………………………………

### BOOKS\_RETURN……..…………………………………………………………………………………………………………….

### COMPENSATION……..…………………………………………………………………………………………………………….

### DIAGRAM overview……..……………………………………………………………………………………………………

## SQL COMMAND

### QUERY USING ORDER BY………………………………………………………………………………………………………

### QUERY USING INNER JOIN……………………………………………………………………………………………………

### QUERY USING AGGREGATE FUNCTIONS…………………………………………………………………………………

### QUERY USING THE GROUP BY AND HAVING CLAUSES……………………………………………………………

### QUERY THAT USES A SUB-QUERY AS A RELATION………………………………………………………………….

### QUERY THAT USES PARTIAL MATCHING IN THE WHERE CLAUSE…………………………………………….

### STORE PROCEDURE……………………………………………………………………………………………………………….

### trigger……………………………………………………………………………………………………………………………….

# INTRODUCE THE PROBLEM

## Describe the problem

Nowaday, libraries were faced with an information explosion and the rapid growth rate of their collections. The issues confronting library administrators during that period were mainly physical managements involving shelving and weeding of materials, storage space, users’ in-house access to the collection, and preservation of the materials. After our team researched, the results are as follows:

* Each person who borrows books needs to have a separate account to manage information instead of checking the physical library account, that information includes necessary personal information such as phone number, address, gender, etc. count, date of birth...
* Each employee of the library also has a separate account to manage the borrowing of books from the library, and has more information of the same staff as the person who borrows the book..
* Books are categorized by publisher and book genre, one producer can produce many books and one genre can have many books.
* The book loan voucher will have a link between an employee and a borrower to represent each time a book is borrowed containing information about the date of borrowing.
* The details of the loan slip will store information about the loan slip, borrowed books, number of books borrowed and return date.

## goals

This database is written to solve the stages and procedures of the library by retrieving information on the computer without the need for manual work as before. This helps both library managers and readers easily control the information of books as well as the return of books in the library.

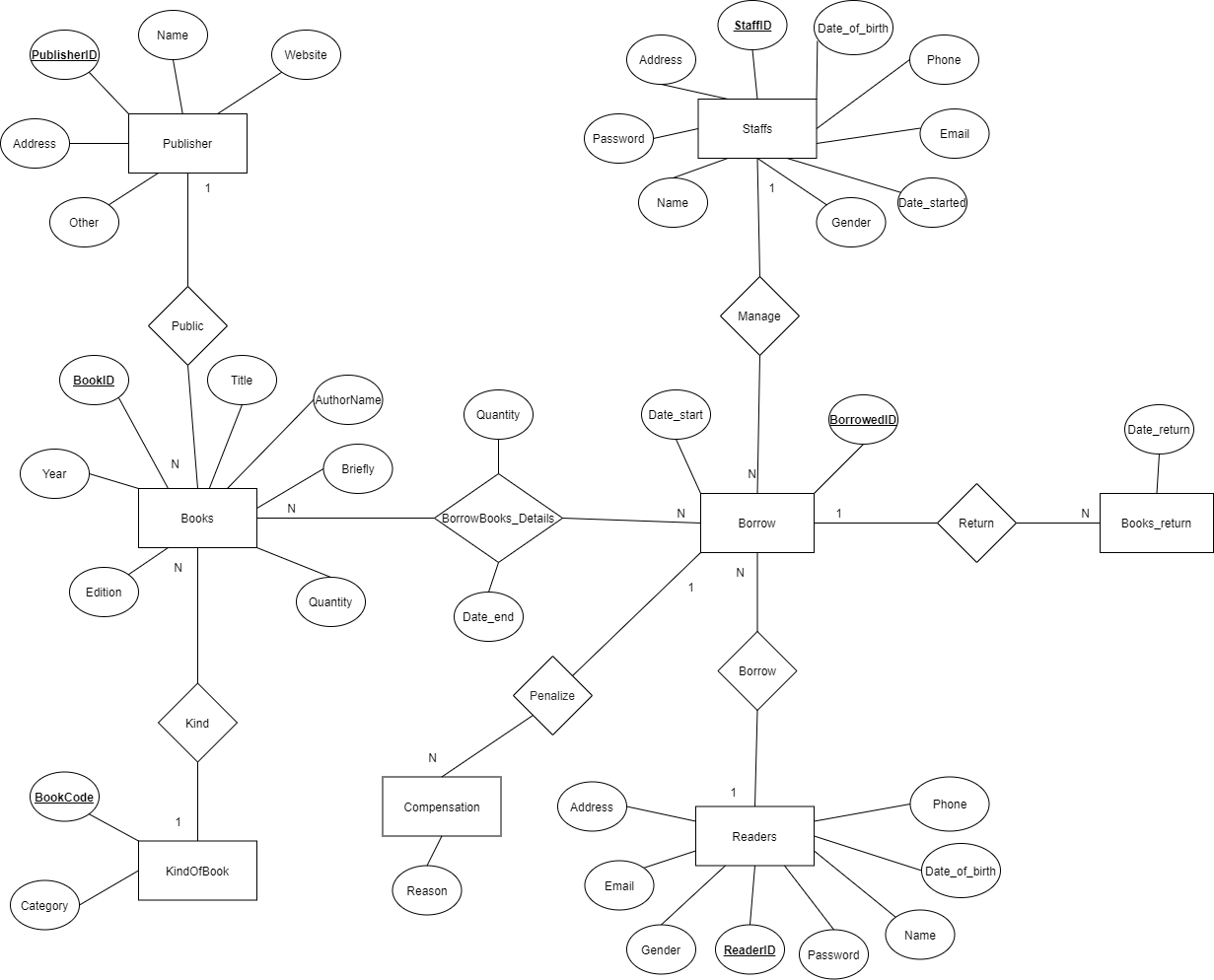
# entity – relationship – erD

## set-up entity – relationship

\* Some symbols used in the model

**Attibute**

|  |  |
| --- | --- |
| * Key / identifier attribute |  |
| * Attribute description / description | **ENTITY**  Attribute |
| * Entity | **WEAK ENTIRY** |
| * Weak entity | *Relationship* |
| * Relationship |  |
| * Connectivity (force) = 1 |  |
| * Connectivity = N |  |



## difinITION entity – attributE

Base on the problem description and management objectives, we can present several entities and attributes of the entity as follow:

* **Publisher**: PublisherID, Name, Address, Website, Other.
* **KindOfBook:** BookCode, Category.
* **Books**: BookID, BookCode, PublisherID, Title, AuthorName, Year, Edition, Quantity, Brief.
* **Staffs**: StaffID, Password, Name, Date\_of\_birth, Address, Gender, Phone, Email, Date-started.
* **Readers**: ReaderID, Password, Name, Date\_of\_birth, Address, Gender, Email.
* **Borrow**: BorrowedID, StaffID, ReaderID, Date\_start.
* **BorrowBooks\_details**: BorrowID,BookID, Quantity, Date\_end.
* **Books\_return**: BorrowID,Date\_return.
* **Compensation**: BorrowedID, Reason.

# database and entity diagram

Just for example on some tables (other table are similar, you have to define all the tables in your database). Note: to run the query you have to define the table 1 first then go to the side tables much

### cREATE DATABASE **LIBRARY\_MANAGEMENT\_SYSTEM**

--create database

CREATE DATABASE LIBRARY\_MANAGEMENT\_SYSTEM

A picture containing table

Description automatically generated

### **Create table publisher**

***Entity - Attribute:***

***Diagram

Description automatically generated***

***Table

Description automatically generated***

***Code:***

--create table Pulisher

CREATE TABLE Publisher(

PublisherID char(50) NOT NULL PRIMARY KEY,

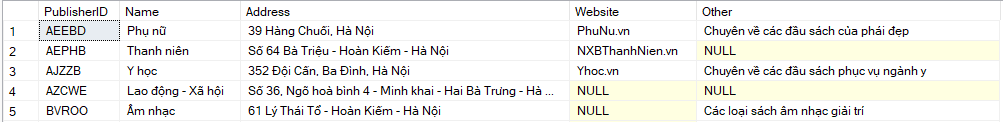
Name nvarchar(200) NOT NULL,

Address nvarchar(200) NOT NULL,

Website char(100),

Other nvarchar(200))

***Example:***



### **Create table kindofbook**

***Entity - Attribute:***

***Diagram

Description automatically generated***

Table

Description automatically generated

***Code:***

--create table KindOfBook

CREATE TABLE KindOfBook(

BookCode char(50) NOT NULL PRIMARY KEY,

Category nvarchar(200) NOT NULL)

***Example:***

Table

Description automatically generated

### **Create table Books**

***Entity - Attribute:***

Diagram

Description automatically generated

Table

Description automatically generated

***Code:***

--create table Books

create table Books(

BookID char(50) NOT NULL PRIMARY KEY,

Title nvarchar(100) NOT NULL,

AuthorName nvarchar(100),

PublisherID char(50) NOT NULL,

BookCode char(50) NOT NULL,

Year int check (Year <= YEAR(getDate())),

Edition int,

Quantity int NOT NULL,

Briefly nvarchar(1000)

constraint fk\_Books\_KindOfBook FOREIGN KEY(BookCode) references KindOfBook(BookCode),

constraint fk\_Books\_Publisher FOREIGN KEY(PublisherID) references Publisher(PublisherID)

)

***Example:***

Graphical user interface, application

Description automatically generated

### Create TABLE **Staffs**

***Entity - Attribute:***

***Diagram

Description automatically generated***

***Code:***

***Graphical user interface, table

Description automatically generated***

--create table Staffs

CREATE TABLE Staffs(

StaffID char(50) NOT NULL PRIMARY KEY CHECK (StaffID like 'FU[0-9][0-9][0-9][0-9][0-9][0-9]'),

Name nvarchar(100) NOT NULL,

Address nvarchar(200),

Date\_of\_birth date NOT NULL check (Date\_of\_birth < getDate()),

Gender bit NOT NULL check (Gender like 0 or Gender like 1),

Phone char(50) UNIQUE CHECK (Phone like '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'),

Email char(100),

Date\_started date NOT NULL,

Password char(50) NOT NULL

)

***Example:***

Graphical user interface

Description automatically generated with low confidence

### create table **readers**

***Entity - Attribute:***

***Diagram

Description automatically generated***

Graphical user interface, application, table

Description automatically generated

***Code:***

--create table Readers

create table Readers(

ReaderID char(50) NOT NULL PRIMARY KEY CHECK (ReaderID like '[A-Z][A-Z][0-9][0-9][0-9][0-9][0-9][0-9]'),

Name nvarchar(100) NOT NULL,

Address nvarchar(200),

Date\_of\_birth date NOT NULL check (Date\_of\_birth < getDate()),

Email char(100),

Gender bit NOT NULL check (Gender like 0 or Gender like 1),

Password char(50) NOT NULL

)

***Example:***

Table

Description automatically generated

### create table **Borrow**

***Entity - Attribute:***

Diagram

Description automatically generated

Table

Description automatically generated

***Code:***

--create table Borrow

create table Borrow(

BorrowID char(50) NOT NULL PRIMARY KEY,

StaffID char(50) NOT NULL,

ReaderID char(50) NOT NULL,

Date\_start date NOT NULL,

constraint fk\_Borrow\_Readers FOREIGN KEY(ReaderID) references Readers(ReaderID),

constraint fk\_Borrow\_Staffs FOREIGN KEY(StaffID) references Staffs(StaffID))

***Example:***

Table

Description automatically generated

### create table **Borrowbooks\_details**

***Entity - Attribute:***

***Diagram

Description automatically generated***

Table

Description automatically generated

***Code:***

--create table BorrowBooks\_details

CREATE TABLE BorrowBooks\_details(

BorrowID char(50) NOT NULL,

BookID char(50) NOT NULL,

Quantity int NOT NULL,

Date\_end date NOT NULL,

constraint fk\_BorrowBooks\_details\_Borrow FOREIGN KEY(BorrowID) references Borrow(BorrowID),

constraint fk\_BorrowBooks\_details\_Books FOREIGN KEY(BookID) references Books(BookID)

primary key(BorrowID,BookID)

)

***Example:***

Table, Excel

Description automatically generated

### Create table **BOOKs\_return**

***Entity - Attribute:***

Diagram, schematic

Description automatically generated

Table

Description automatically generated

***Code:***

--create table Books\_return

CREATE TABLE Books\_return(

BorrowID char(50) NOT NULL,

Date\_return date NOT NULL,

constraint fk\_Books\_return\_Borrow FOREIGN KEY(BorrowID) references Borrow(BorrowID))

***Example:***

A picture containing table

Description automatically generated

### create table **Compensation**

***Entity - Attribute:***

Diagram

Description automatically generated

Table

Description automatically generated

***Code:***

--create table Compensation

create table Compensation(

BorrowID char(50) NOT NULL,

Reason nvarchar(200) NOT NULL,

constraint fk\_PHAT\_MUONSACH FOREIGN KEY(BorrowID) references Borrow(BorrowID))

***Example:***

Graphical user interface, application, table

Description automatically generated



# V. sql command

I using Microsoft SQL Server 2019, this server build intelligent, mission-critical applications using a scalable, hybrid database platform that has everything built in—from in-memory performance and advanced security to in-database analytics.

## query using order by

***Code:***

SELECT \* FROM Staffs s

ORDER BY s.Name

***Result:***

A picture containing graphical user interface

Description automatically generated

We use query containing ORDER BY to sort the list ascending or descending by the values of a domain. SELECT \* FROM Staffs command give us all record in Staffs table and then sort the records ascending by name

## query using inner join

***Code:***

SELECT \* FROM Books b

inner join Publisher p

ON b.PublisherID = p.PublisherID

WHERE b.Year = 2018

***Result:***

Table

Description automatically generated

We use **INNER JOIN** to select book publish in 2018 and its website

## query using aggregate functions

***Code:***

SELECT COUNT(B.BookID) FROM Books B, Publisher P

WHERE B.PublisherID = P.PublisherID

AND P.Name = N'Kim Đồng'

***Result:***



We use function **COUNT()** with parameter BookID to count the number of book has publisher name is Kim Đồng

***Code:***

SELECT TOP(5) \* FROM Books

ORDER BY Quantity DESC

***Result:***

Graphical user interface, application, chat or text message

Description automatically generated

We use function **TOP()** to select top 5 with the most number of books

## query using the group by and having clauses

***Code:***

SELECT S.StaffID, S.Name, COUNT(B.BorrowID) AS TOTALMANAGE FROM Staffs S, Borrow B

WHERE S.StaffID = B.StaffID

GROUP BY S.StaffID, S.Name

HAVING COUNT(B.BorrowID) > 3

***Result:***

Graphical user interface, text, application, table

Description automatically generated

We use **GROUP** BY and **HAVING** clauses to count staffs manage more than 3 bills

## query that uses a sub-query as a relation

***Code1:***

SELECT b.BookID,

b.Title,

count(br.ReaderID) AS TotalBorrows

FROM Books b,

BorrowBooks\_details bd,

Borrow br,

Readers r

WHERE b.BookID = bd.BookID

AND bd.BorrowID = br.BorrowID

AND br.ReaderID = r.ReaderID

GROUP BY b.BookID,

b.Title

HAVING count(br.BorrowID) =

(SELECT top(1) count(br.ReaderID) AS TotalBorrows

FROM Books b,

BorrowBooks\_details bd,

Borrow br,

Readers r

WHERE b.BookID = bd.BookID

AND bd.BorrowID = br.BorrowID

AND r.ReaderID = br.ReaderID

GROUP BY b.BookID,

b.Title

ORDER BY count(br.BorrowID) DESC)

***Result1:***

Graphical user interface, text, application, email

Description automatically generated

We using subquery to find the most book borrowed by reader and get in into having clause

***Code2:***

SELECT r.ReaderID,

r.Name,

sum(bd.Quantity) AS NumberOfBorrowedBooks

FROM Readers r,

Borrow b,

BorrowBooks\_details bd

WHERE r.ReaderID = b.ReaderID

AND b.BorrowID = bd.BorrowID

GROUP BY r.ReaderID, r.Name

HAVING sum(bd.Quantity) =

(SELECT top(1) sum(bd.Quantity) AS NumberOfBorrowedBooks

FROM Readers r,

Borrow b,

BorrowBooks\_details bd

WHERE r.ReaderID = b.ReaderID

AND b.BorrowID = bd.BorrowID

GROUP BY r.ReaderID,

r.Name,

bd.BorrowID

ORDER BY sum(bd.Quantity) DESC)

***Result2:***

Graphical user interface, application

Description automatically generated

We use subquery to find the reader borrowed most book and get it into having clause

## query that uses partial matching in the where clause

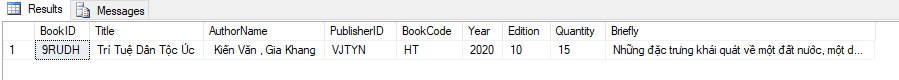
***Code:***

SELECT \*

FROM Books

WHERE Title like N'%Trí Tuệ%' and Quantity > 10

***Result:***



We use PARTIAL MATCHING in the WHERE clauses to combine binding conditions

## store procedure

***Code:***

CREATE PROC check\_Quantity @Book\_ID CHAR(10), @NumberofBooks INT OUTPUT

AS

BEGIN

SET @NumberofBooks = (SELECT Quantity

FROM Books WHERE BookID = @Book\_ID)

END

DECLARE @t INT

EXEC check\_Quantity'054JD', @t OUTPUT

SELECT @t AS Quantity

***Result:***

Graphical user interface, application

Description automatically generated

We use procedure to check the quantity of books for which the book's ID is entered by the librarian or user.

## trigger

***Code1:***

CREATE TRIGGER checkCompensation

ON Books\_return AFTER INSERT

AS

BEGIN

DECLARE @borrowID CHAR(50);

DECLARE @date\_end INT

SELECT @borrowID = i.BorrowID, @date\_end = DATEDIFF(day,bd.Date\_end,i.Date\_return) FROM inserted i,

(SELECT DISTINCT bd.BorrowID, bd.Date\_end FROM BorrowBooks\_details bd) AS bd

WHERE bd.BorrowID = i.BorrowID

IF(@date\_end >0)

BEGIN

INSERT INTO Compensation(BorrowID,Reason) VALUES

(@borrowID, N'Bị nộp muộn ' + CONVERT(NVARCHAR(200),@date\_end) + N' ngày')

END

END

***Result1:***

Graphical user interface, application, table

Description automatically generated

Add overdue loan slip to the table compensation

***Code2:***

CREATE TRIGGER checkRemainQuanity

ON BorrowBooks\_details AFTER INSERT

AS

BEGIN

DECLARE @quantity INT;

SELECT @quantity = (SELECT b.Quantity - (SELECT sum(b.Quantity) FROM inserted i, BorrowBooks\_details b

WHERE i.BookID = b.BookID)

FROM Books b, inserted i WHERE b.BookID = i.BookID)

IF(@quantity<0)

BEGIN

PRINT 'Not enough quantity'

DELETE FROM BorrowBooks\_details

where Quantity = (SELECT i.Quantity FROM inserted i) and

BorrowID = (SELECT i.BorrowID FROM inserted i)

and BookID = (SELECT i.BookID FROM inserted i)

END

END

**Check to see if the quantity of books is enough for the borrower**

***Result2:***

Graphical user interface, application

Description automatically generated

THE END