DBI202 - Final Project

Book and Stationery Management System Database

Tran Thanh Duong - Nguyen Tan Loc

GitHub

TABLE OF CONTENTS

Introduction

Problem description details

The **FAHASA Company** needs to build a database to manage books and stationeries as well as employees and customers. After the actual investigation, the results are as follows:

- A company has many branches.
- Every branch will have a manager who is also an employee at that branch, keeping track of the start date and end date of every manager.
- Employees will be attended every slot in days and every day in weeks.
- Every bookstore will manage two main products: book and stationery. A product will have common information of two above kinds of products. There is separate additional information for each one.
- Every bookstore manages the amount of each product.
- Each bookstore needs to have their own bill: offline bill for buying directly at the bookstore and online bill for buying on website, but these bills must be reported to the main bill table.
- Customers can buy a product from the company via two ways: online and offline.
- Customers who want to buy via online transaction must have an account in the system. Their accounts can have multiple addresses but only one delivery address in an online bill.
- But for customers who buy directly at the bookstore (offline customers), we can collect some of their information such as their name, their phone number and their address; this information is not compulsory. In case offline customers provide a phone number, we can use it to derive their accounts if they have already registered.
- Bill can get paid by some payment methods such as credit card, on delivery, electronic wallet (MOMO, ViettelPay, etc.), every bill has its own payment code for sending requests to online banking or electronic wallet through API.
- Customer accounts will have basic information about customers, which are managed directly by the company's system and used to help the customer service team.
- In case a customer buys a product directly from the company's stores, there will be an employee directly working on this customer and taking charge of the bill of this customer.
- The company provides many attractive discount programs for their products and vouchers for bills:
    - Discount programs are applied for products only, the price of products will be reduced due to discount percentage, but they can't be reduced over the limit.

    - Vouchers are applied for bills, the invoices will be reduced due to voucher percentage, but they can't be reduced over the limit and are applied if those invoices reach the required amount of money. In vouchers, there is also free shipping included or not and the number of remaining uses of that voucher.
- Every product can only apply to one discount program only. An user can store multiple vouchers but they can apply one voucher only for each bill and has a limited number of uses for that voucher.
- The information of all transactions will be stored and can be looked up later on. Bill of online transactions, and offline transactions will be stored separately, but they are reported to a main bill table.

Request of the company

- Identify stores with high sales and stores with low sales to have a reasonable promotion strategy.
- Find out which products are bestsellers, and also low-profit products.
- Provide better customer service (i.e, giving some special vouchers) for those who frequently buy through membership rank.
- Find the favorite authors based on revenue from those authors' books.

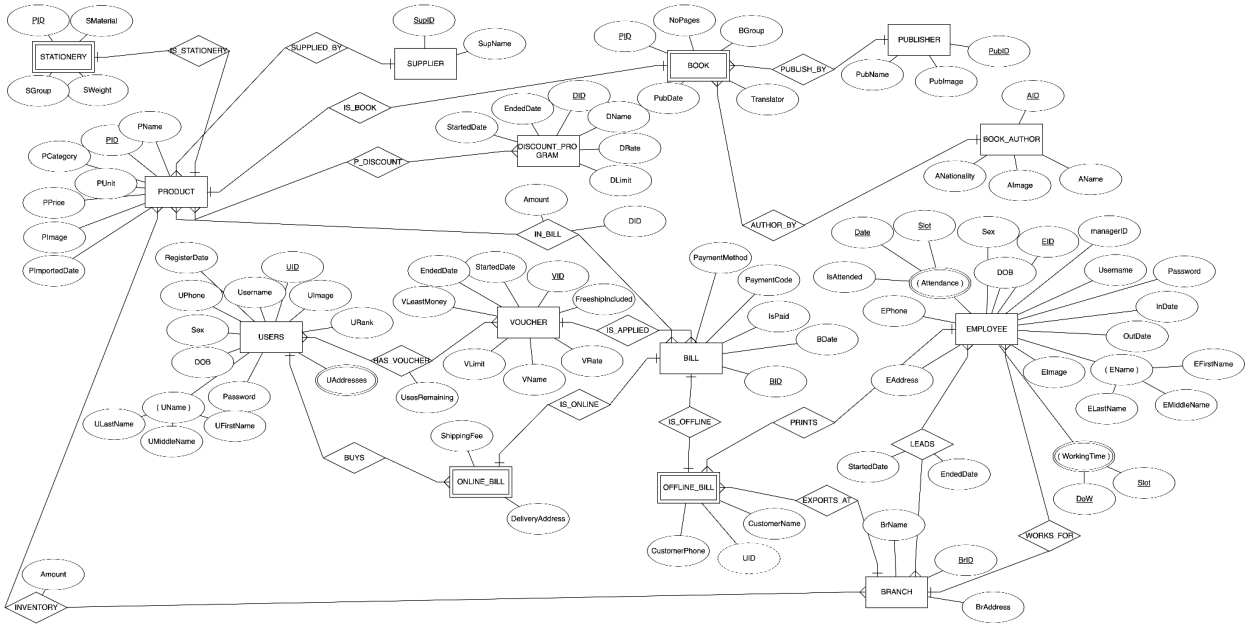
Entity Relationship Model

Entity identifying - Attribute

- **BRANCH: branch ID**, name, address
- **SUPPLIER: supplier ID**, supplier name
- **PRODUCT product ID**, category, unit, imported date, product name, price, image, supplier ID
- **BOOK_AUTHOR**: **author ID**, name, image, nationality
- **PUBLISHER**: **publisher ID** , name , image
- **BOOK**: **product ID** , publisher date, translator , number of pages , group , author ID , publisher ID
- **STATIONERY**: **product ID** , weight, group , material
- **DISCOUNT_PROGRAM**: **discount ID**, name , rate, limit , started date, ended date
- **P_DISCOUNT**: **product ID** , **discount ID**
- **VOUCHER**: **voucher ID** , name , rate, limit, least money, started date, ended date, free shipping included
- **USER**: **user ID**, username , password , first name , middle name, last name , DOB, Sex , phone , image , rank , register date
- **HAS_VOUCHER**: **user ID** , **voucher ID** , uses remaining

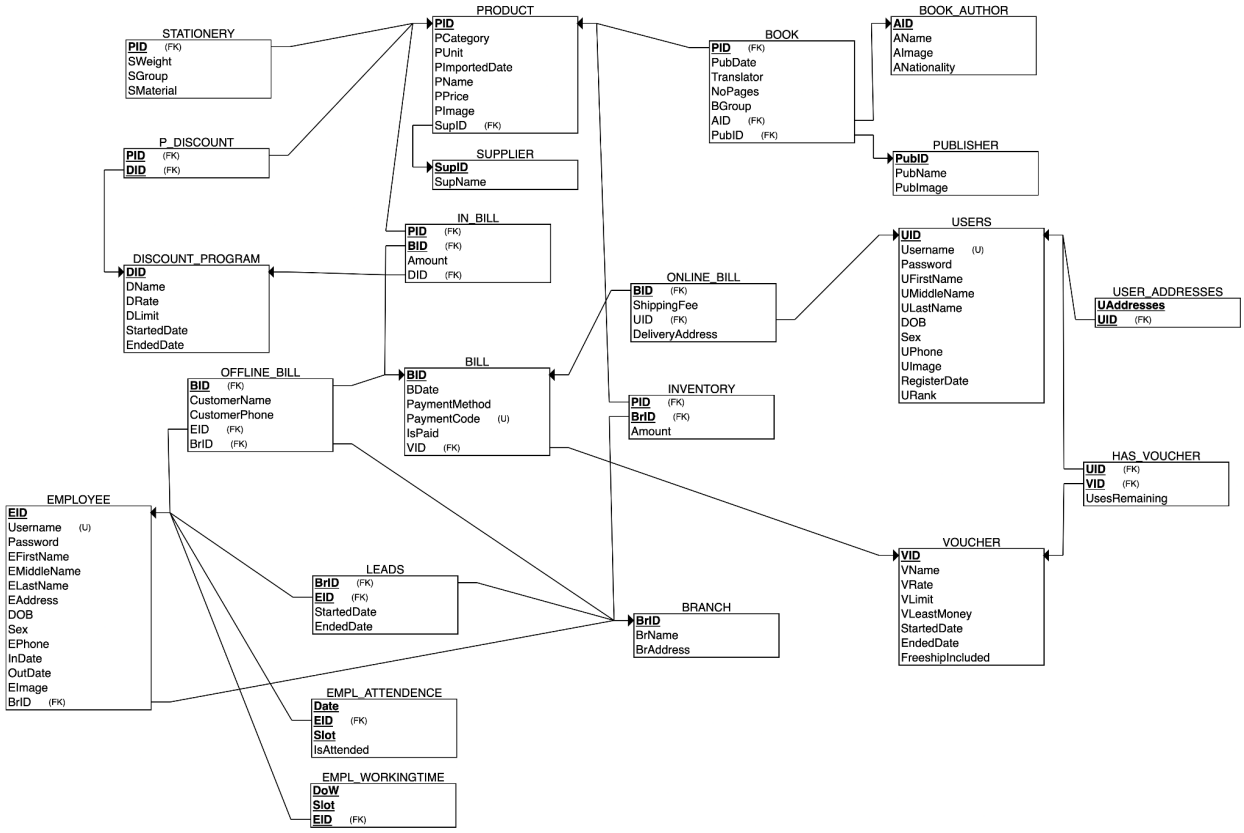
- **USER_ADDRESSES**: **user ID** , **address**
- **EMPLOYEE**: **employee ID** , username , password , first name , middle name , last name , address , DOB, sex , phone , in-date, out-date, image , branch ID
- **EMPL_WORKINGTIME**: **day of week** , **slot** , **employee ID**
- **EMPL_ATTENDENCE**: **date**, **slot** , **employee ID** , is attended
- **BILL**: **bill ID**, date, payment method , payment code , is paid , voucher ID
- **ONLINE_BILL**: **bill ID** , shipping fee , user ID , delivery address
- **OFFLINE_BILL**: **bill ID** , customer name , customer phone, employee ID , branch ID
- **IN_BILL**: **product ID** , **bill ID** , amount , discount ID
- **INVENTORY**: **product ID** , **branch ID** , amount
- **LEADS**: **branch ID** , **employee ID** , started date, ended date

Entity Relationship diagram



ERD - Preview

Relational Database



[Relational Database - Preview]

Data Dictionary

Defining Tables

BRANCH Table

a. Field definition

Field name	Data type	Default	Check	Key/Index/Constraint
BrID	INTEGER			PK
BrName	NVARCHAR(50)			NOT NULL
BrAddress	NVARCHAR(255)			NOT NULL

b. Example

BrID	BrName	BrAddress
1	FAHASA Sài Gòn	60-62 Lê Lợi, P. Bến Nghé, Q. 1, TP. HCM
2	FAHASA Tân Bình	364 Trường Chinh, P 13, Q. Tân Bình, TP.HCM
3	FAHASA Hà Nội	338 Xã Đàn, P. Phương Liên, Q. Đống Đa, Hà Nội
4	FAHASA Buôn Mê Thuột	71 Nguyễn Tất Thành, P. Tân An, TP.Buôn Ma Thuột, Đăk Lăk
5	FAHASA Đà Nẵng	Số 300-302 Lê Duẩn, P. Tân Chính, Q. Thanh Khê, TP. Đà Nẵng
6	FAHASA An Giang	Lầu 3, Số 12 Nguyễn Huệ A ,P. Mỹ Long ,TP.Long Xuyên , An Giang

c. Code

```
CREATE TABLE BRANCH
(
  BrID INT NOT NULL,
  BrAddress NVARCHAR(255) NOT NULL,
  BrName NVARCHAR(50) NOT NULL,
  PRIMARY KEY (BrID)
```

```
);
```

SUPPLIER Table

a. Field definition

Field name	Data type	Default	Check	Key/Index/Constraint
SupID	INTEGER			PK
SupName	NVARCHAR(50)			NOT NULL

b. Example

SupID	SupName
1	Minh Long Book
2	Đinh Tị Books
3	Nhã Nam
4	Thiên Long
5	Vận Tải Quốc Anh

c. Code

```
CREATE TABLE SUPPLIER
(
  SupID INT NOT NULL,
  SupName NVARCHAR(50) NOT NULL,
  PRIMARY KEY (SupID)
);
```

PRODUCT Table

a. Field definition

Field name	Data type	Default	Check	Key/Index/Constraint
PID	INTEGER			PK
PCategory	VARCHAR(20)		PCategory IN ('book', 'stationery')	NOT NULL

PUnit	VARCHAR(10)			NOT NULL
PImportedDate	DATETIME		<= GETDATE()	NOT NULL
PName	NVARCHAR(255)			NOT NULL
PPrice	INTEGER		>= 0	NOT NULL
PImage	VARCHAR(255)			
SupID	INTEGER			FK, references to SUPPLIER(SupID)

b. Example

PID	**PCategory**	**PUnit**	**PImportedDate**	**PName**	**PPrice**	**PImage**	**SupID**
1	book	item	1/1/2021	Nhà Giả Kim (Tái bản 2020)	79000	./images/products/books/nha-gia-kim-2021.png	3
2	book	collection	6/12/2020	Combo Truyện Cổ Tích Việt Nam Dành Cho Thiếu Nhi: Thạch Sanh + Thánh Gióng + Tấm Cám (Bộ 3 Cuốn)	30000	./images/products/books/combo-truyen-co-tich-viet-nam-danh-cho-thieu-nhi.png	1
3	stationery	collection	1/11/2021	Bộ 6 Bút Lông Kim Màu Pastel Marvy 4300	135000	./images/products/stationeries/bo-6-but-long-kim-mau-pastel.png	5
4	stationery	item	5/8/2021	Bút Chì Bấm 0.5mm Thiên Long	18500	./images/products/stationeries/	4

				PC-024 - Màu Xanh		but-chi-bam-05mm-thien-long.png	

c. Code

```
CREATE TABLE PRODUCT
(
  PID INT NOT NULL,
  PCategory VARCHAR(20) NOT NULL,
  PUnit VARCHAR(10) NOT NULL,
  PImportedDate DATETIME NOT NULL,
  PName NVARCHAR(255) NOT NULL,
  PPrice INTEGER NOT NULL,
  PImage VARCHAR(255),
  SupID INT NOT NULL,
  PRIMARY KEY (PID),
  FOREIGN KEY (SupID) REFERENCES SUPPLIER(SupID),
  CONSTRAINT chk_cate CHECK (PCategory IN ('book', 'stationery')),
  CONSTRAINT chk_price CHECK (PPrice >= 0),
  CONSTRAINT chk_prodate CHECK (PImportedDate <= GETDATE())

);
```

BOOK_AUTHOR Table

a. Table definition

Field name	Data type	Default	Check	Key/Index/Constraint
AID	INTEGER			PK
AName	NVARCHAR(50)			NOT NULL
AImage	VARCHAR(255)			
ANationality	VARCHAR(15)			

b. Example

AID	AName	AImage	ANationality
1	Paulo Coelho	./images/authors/paulo-coelho.jpeg	Brazilian
2	Minh Long		Vietnam

c. Code

```
CREATE TABLE BOOK_AUTHOR
(
  AID INT NOT NULL,
  AName NVARCHAR(50) NOT NULL,
  AImage VARCHAR(255),
  ANationality VARCHAR(15),
  PRIMARY KEY (AID)
);
```

PUBLISHER Table

a. Table definition

Field name	Data type	Default	Check	Key/Index/Constraint
PubID	INTEGER			PK
PubName	NVARCHAR(50)			NOT NULL
PubImage	VARCHAR(255)			

b. Example

PubID	PubName	PubImage
1	NXB Hội Nhà Văn	./images/publishers/hoi-nha-van.png
2	NXB Văn Học	./images/publishers/van-hoc.png

c. Code

```
CREATE TABLE PUBLISHER
(
  PubID INT NOT NULL,
```

```
  PubName NVARCHAR(50) NOT NULL,
  PubImage VARCHAR(255),
  PRIMARY KEY (PubID)
);
```

BOOK Table

a. Table definition

Field name	Data type	Default	Check	Key/Index/Constraint
PID	INTEGER			PK, FK, references to PRODUCT(PID)
PubDate	DATE		<= GETDATE()	NOT NULL
Translator	NVARCHAR(50)			
NoPages	INTEGER		> 0	NOT NULL
BGroup	VARCHAR(20)			
AID	INTEGER			FK, references to AUTHOR(AID)
PubID	INTEGER			FK, references to PUBLISHER(PubID)

b. Example

PID	PubDate	Translator	NoPages	BGroup	AID	PubID
1	1/11/2020	Lê Chu Cầu	227	literature	1	1
2	15/11/2020			comic	2	2

c. Code

```
CREATE TABLE BOOK
(
  PubDate DATE NOT NULL,
  Translator NVARCHAR(50),
```

```
  NoPages INT NOT NULL,
  BGroup VARCHAR(20),
  AID INT NOT NULL,
  PubID INT NOT NULL,
  PID INT NOT NULL,
  PRIMARY KEY (PID),
  FOREIGN KEY (AID) REFERENCES BOOK_AUTHOR(AID),
  FOREIGN KEY (PubID) REFERENCES PUBLISHER(PubID),
  FOREIGN KEY (PID) REFERENCES PRODUCT(PID),
  CONSTRAINT chk_nopages CHECK (NoPages > 0),
  CONSTRAINT chk_bookdate CHECK (PubDate <= GETDATE())
);
```

STATIONERY Table

a. Table definition

Field name	Data type	Default	Check	Key/Index/Constraint
PID	INTEGER			PK, FK, references to PRODUCT(PID)
SWeight	FLOAT		> 0	NOT NULL
SGroup	VARCHAR(20)			
SMaterial	VARCHAR(20)			

b. Example

PID	SWeight	SGroup	SMaterial
3	50	pen	plastic
4	12	pen	plastic

c. Code

```
CREATE TABLE STATIONERY
(
  SWeight FLOAT NOT NULL,
  SGroup VARCHAR(20),
  SMaterial VARCHAR(20),
  PID INT NOT NULL,
  PRIMARY KEY (PID),
```

```
  FOREIGN KEY (PID) REFERENCES PRODUCT(PID),
  CONSTRAINT chk_weight CHECK (SWeight > 0.0)
);
```

DISCOUNT_PROGRAM Table

a. Table definition

Field name	Data type	Default	Check	Key/Index/Constraint
DID	INTEGER			PK
DName	NVARCHAR(100)			
DRate	FLOAT		DRate BETWEEN 0.0 AND 1.0	NOT NULL
DLimit	INTEGER		>= 0	NOT NULL
StartedDate	DATE			NOT NULL
EndedDate	DATE			NOT NULL

b. Example

DID	DName	DRate	DLimit	StartedDate	EndedDate
1	Mừng ngày Phụ Nữ Việt Nam 20/10 (2021)	0.2	50000	20/10/2021	1/11/2021
2	Mừng ngày Nhà Giáo Việt Nam 20/11 (2021)	0.3	75000	20/11/2021	21/11/2021

c. Code

```
CREATE TABLE DISCOUNT_PROGRAM
(
  DID INT NOT NULL,
  DName NVARCHAR(100) NOT NULL,
  DRate FLOAT NOT NULL,
```

```
  DLimit INT NOT NULL,
  StartedDate DATE NOT NULL,
  EndedDate DATE,
  PRIMARY KEY (DID),
  CONSTRAINT chk_Rate CHECK (DRate BETWEEN 0.0 AND 1.0),
  CONSTRAINT chk_limit CHECK (DLimit >= 0)
);
```

P_DISCOUNT Table

a. Table definition

Field name	Data type	Default	Check	Key/Index/Constraint
PID	INTEGER			PK, FK, references to PRODUCT(PID)
DID	INTEGER			PK, FK, references to DISCOUNT_PROGRAM(DID)

b. Example

PID	DID
3	1
4	1
1	2
2	2

c. Code

```
CREATE TABLE P_DISCOUNT
(
  PID INT NOT NULL,
  DID INT NOT NULL,
  PRIMARY KEY (PID, DID),
  FOREIGN KEY (PID) REFERENCES PRODUCT(PID),
  FOREIGN KEY (DID) REFERENCES DISCOUNT_PROGRAM(DID)
);
```

VOUCHER Table

a. Table definition

Field name	Data type	Default	Check	Key/Index/Constraint
VID	INTEGER			PK
VName	NVARCHAR(100)			
VRate	FLOAT		VRate BETWEEN 0.0 AND 1.0	
VLimit	INTEGER		>= 0	NOT NULL
VLeastMoney	INTEGER		>= 0	NOT NULL
StartedDate	DATE			NOT NULL
EndedDate	DATE			
FreeshipIncluded	BIT			

b. Example

VID	VName	VRate	VLimit	VLeastMoney	StartedDate	EndedDate	FreeshipIncluded
1	[HALLOWEEN] Giảm tối đa 5% tối đa 20K cho đơn hàng từ 100K	0.05	20000	100000	27/10/2021	31/10/2021	0
2	[Ưu đãi khách VIP] Giảm 10% tối đa 50K cho đơn từ 150K	0.1	50K	150000	5/11/2021	12/11/2021	1

c. Code

```
CREATE TABLE VOUCHER
(
  VID INT NOT NULL,
  VName NVARCHAR(100) NOT NULL,
  VRate FLOAT NOT NULL,
  VLimit INT NOT NULL,
  VLeastMoney INT NOT NULL,
  StartedDate DATE NOT NULL,
  EndedDate DATE,
  FreeshipIncluded BIT NOT NULL,
  PRIMARY KEY (VID),
  CONSTRAINT chk_vrate CHECK (VRate BETWEEN 0.0 AND 1.0),
  CONSTRAINT chk_vlimit CHECK(VLimit >= 0),
  CONSTRAINT chk_vleastmoney CHECK(VLeastMoney >= 0)
);
```

USERS Table

a. Table definition

Field name	Data type	Default	Check	Key/Index/Constraint
UID	INTEGER			PK
Username	VARCHAR(50)			UNIQUE
Password	VARCHAR(255)			
UFirstName	NVARCHAR(10)			
UMiddleName	NVARCHAR(30)			
ULastName	NVARCHAR(10)			
DOB	DATE		<= GETDATE()	
Sex	VARCHAR(1)		Sex IN ('F', 'M')	
UPhone	VARCHAR(15)			
UImage	VARCHAR(255)			

URank	VARCHAR(50)	newbie	URank IN ('brown', 'silver', 'gold', 'diamond', ' newbie')	NOT NULL
RegisterDate	DATETIME		<= GETDATE()	NOT NULL

b. Example

UID	Username	Password	UFirstName	UMiddleName	ULastName	DOB	Sex	UPhone	UImage	URank	RegisterDate
1	user1	abc@xyz	Nguyễn	Văn	A	1/1/2002	M	123456789		silver	8/3/2020
2	user2	abcd@123	Trần	Thị	B	2/3/1998	F	987654321	./images/users/user2/profile-image.jpg	gold	5/5/2019

c. Code

```
CREATE TABLE USERS
(
  UID INT NOT NULL,
  Username VARCHAR(50) NOT NULL,
  Password VARCHAR(255) NOT NULL,
  UFirstName NVARCHAR(10),
  UMiddleName NVARCHAR(30),
  ULastName NVARCHAR(10),
  DOB DATE,
  Sex VARCHAR(1),
  UPhone VARCHAR(15),
  UImage VARCHAR(255),
  RegisterDate DATETIME NOT NULL,
  URank VARCHAR(50) NOT NULL,
  PRIMARY KEY (UID),
  UNIQUE (Username),
  CONSTRAINT chk_usex CHECK (Sex IN ('F' ,'M')),
```

```
  CONSTRAINT chk_rank CHECK (URank IN ('brown', 'silver', 'gold',
'diamond', 'newbie')),
  CONSTRAINT chk_udate CHECK (DOB <= GETDATE() AND RegisterDate <=
GETDATE())
);
```

HAS_VOUCHER Table

a. Table definition

Field name	Data type	Default	Check	Key/Index/Constraint
UID	INTEGER			PK, FK, references to USER(UID)
VID	INTEGER			PK, FK, references to VOUCHER(VID)
UsesRemaining	INTEGER	1	>= 0	NOT NULL

b. Example

UID	VID	UsesRemaining
1	1	1
2	2	3

c. Code

```
CREATE TABLE HAS_VOUCHER
(
  UsesRemaining INT NOT NULL,
  VID INT NOT NULL,
  UID INT NOT NULL,
  PRIMARY KEY (VID, UID),
  FOREIGN KEY (VID) REFERENCES VOUCHER(VID),
  FOREIGN KEY (UID) REFERENCES USERS(UID),
  CONSTRAINT chk_usesremaining CHECK (UsesRemaining >= 0)
);
```

USER_ADDRESSES Table

a. Table definition

Field name	Data type	Default	Check	Key/Index/Constraint
UID	INTEGER			PK, FK, references to USER(UID)
UAddresses	NVARCHAR(255)			PK

b. Example

UID	UAddresses
1	Phường Long Thạnh Mỹ, Thành Phố Thủ Đức, TP.HCM
1	TP. Long Xuyên, Tỉnh An Giang
2	Phường Phước Long B, Thành Phố Thủ Đức, TP.HCM
2	P7, Thành phố Bạc Liêu, Tỉnh Bạc Liêu

c. Code

```
CREATE TABLE USER_ADDRESSES
(
  UAddresses NVARCHAR(255) NOT NULL,
  UID INT NOT NULL,
  PRIMARY KEY (UAddresses, UID),
  FOREIGN KEY (UID) REFERENCES USERS(UID)
);
```

EMPLOYEE Table

a. Table definition

Field name	Data type	Default	Check	Key/Index/Constraint
EID	INTEGER			PK
Username	VARCHAR(50)			UNIQUE
Password	VARCHAR(255)			NOT NULL

EFirstName	NVARCHAR(10)			NOT NULL
EMiddleName	NVARCHAR(30)			NOT NULL
ELastName	NVARCHAR(10)			NOT NULL
EAddress	NVARCHAR(255)			NOT NULL
DOB	DATE		<= GETDATE()	NOT NULL
Sex	VARCHAR(1)		Sex in ('F', 'M')	NOT NULL
EPhone	VARCHAR(15)			NOT NULL
InDate	DATETIME		<= GETDATE()	NOT NULL
OutDate	DATETIME			
EImage	VARCHAR(255)			NOT NULL
BrID	INTEGER			FK, references to BRANCH(BrID)

b. Example

EID	Username	Password	EFirst Name	EMiddleName	ELast Name	EAddress	DOB	Sex	EPhone	InDate	OutDate	BrID
1	emp2	abcxyz	Trần	Thiên	B		3/3/1999	M	2222222	5/8/2020		2
2	emp1	abc@123	Đào	Bá	A		14/2/2000	M	11111111	12/3/2019		1

c. Code

```
CREATE TABLE EMPLOYEE
(
  EID INT NOT NULL,
  Username VARCHAR(50) NOT NULL,
  Password VARCHAR(255) NOT NULL,
  EFirstName NVARCHAR(10) NOT NULL,
  EMiddleName NVARCHAR(30) NOT NULL,
```

```
    ELastName NVARCHAR(10) NOT NULL,
    EAddress NVARCHAR(255) NOT NULL,
    DOB DATE NOT NULL,
    Sex VARCHAR(1) NOT NULL,
    EPhone VARCHAR(15) NOT NULL,
    InDate DATETIME NOT NULL,
    OutDate DATETIME,
    EImage VARCHAR(255) NOT NULL,
    BrID INT NOT NULL,
    PRIMARY KEY (EID),
    FOREIGN KEY (BrID) REFERENCES BRANCH(BrID),
    UNIQUE (Username),
    CONSTRAINT chk_sex CHECK (Sex IN ('F', 'M')),
    CONSTRAINT chk_empldate CHECK (DOB <= GETDATE() and InDate <= GETDATE())

);
```

EMPL_WORKINGTIME Table

a. Table definition

Every employee will choose to work in some specific dates in a week, the DoW field is accepted the definition of Days Of Week Code from this website: https://registrar.gmu.edu/topics/days-of-the-week-codes/

Field name	Data type	Default	Check	Key/Index/Constraint
DoW	VARCHAR(1)		DoW IN ('M', 'T', 'W', 'R', 'F', 'S', 'U')	PK
Slot	INTEGER		> 0 AND <= 10	PK
EID	INTEGER			PK, FK, references to EMPLOYEE(EID)

b. Example

DoW	Slot	EID
M	1	1

W	8	1
F	8	1
T	2	2
R	3	2
S	2	2

c. Code

```
CREATE TABLE EMPL_WORKINGTIME
(
  DoW VARCHAR(1) NOT NULL,
  Slot INT NOT NULL,
  EID INT NOT NULL,
  PRIMARY KEY (DoW, Slot, EID),
  FOREIGN KEY (EID) REFERENCES EMPLOYEE(EID),
  CONSTRAINT chk_dow CHECK (DoW IN ('M', 'T', 'W', 'R', 'F', 'S', 'U')),
  CONSTRAINT chk_slot CHECK (Slot > 0 AND Slot <= 10)
);
```

EMPL_ATTENDENCE Table

a. Table definition

The Date field will be generated depending on the Days of Weeks that employee chooses.

Field name	Data type	Default	Check	Key/Index/Constraint
Date	DATE			PK
Slot	INTEGER			PK
EID	INTEGER			PK, FK, references to EMPLOYEE(EID)
IsAttended	BIT			

b. Example

Date	Slot	EID	IsAttended
25/10/2021	1	1	1
27/10/2021	8	1	1

29/10/2021	8	1	1
26/10/2021	2	2	1
28/10/2021	3	2	0
30/10/2021	2	2	1

c. Code

```
CREATE TABLE EMPL_ATTENDENCE
(
  Date DATE NOT NULL,
  Slot INT NOT NULL,
  IsAttended BIT,
  EID INT NOT NULL,
  PRIMARY KEY (Date, Slot, EID),
  FOREIGN KEY (EID) REFERENCES EMPLOYEE(EID)
);
```

BILL Table

a. Table definition

Field name	Data type	Default	Check	Key/Index/Constraint
BID	INTEGER			PK
BDate	DATETIME		<= GETDATE()	NOT NULL
PaymentMethod	VARCHAR(15)		PaymentMethod IN ('cash'', 'credit card', 'MOMO', 'ViettelPay')	NOT NULL
PaymentCode	VARCHAR(50)			UNIQUE
IsPaid	BIT			NOT NULL
VID	INTEGER			FK, references to VOUCHER(VID)

b. Example

BID	BDate	PaymentMethod	PaymentCode	IsPaid	VID
1	1/11/2021	cash	CB2EEGDE	0	
2	5/11/2021	MOMO	HG2EKQ9J	1	2

c. Code

```
CREATE TABLE BILL
(
  BID INT NOT NULL,
  BDate DATETIME NOT NULL,
  PaymentMethod VARCHAR(15) NOT NULL,
  PaymentCode VARCHAR(50) NOT NULL,
  IsPaid BIT NOT NULL,
  VID INT NOT NULL,
  PRIMARY KEY (BID),
  FOREIGN KEY (VID) REFERENCES VOUCHER(VID),
  UNIQUE (PaymentCode),
  CONSTRAINT chk_paymentmethod CHECK (PaymentMethod IN ('cash', 'credit
card', 'MOMO', 'ViettelPay')),
  CONSTRAINT chk_billdate CHECK (BDate <= GETDATE())
);
```

ONLINE_BILL Table

a. Table definition

Field name	Data type	Default	Check	Key/Index/Constraint
BID	INTEGER			PK, FK, references to BILL(BID)
ShippingFee	INTEGER		>= 0	NOT NULL
DeliveryAddress	NVARCHAR(255)			NOT NULL
UID	INTEGER			FK, references to USER(UID)

b. Example

BID	ShippingFee	DeliveryAddress	UID
1	23000	Phường Long Thạnh Mỹ, Thành Phố Thủ Đức, TP.HCM	1

c. Code

```
CREATE TABLE ONLINE_BILL
(
  DeliveryAddress NVARCHAR(255) NOT NULL,
  ShippingFee INT NOT NULL,
  UID INT NOT NULL,
  BID INT NOT NULL,
  PRIMARY KEY (BID),
  FOREIGN KEY (UID) REFERENCES USERS(UID),
  FOREIGN KEY (BID) REFERENCES BILL(BID),
  CONSTRAINT chk_shipfee CHECK (ShippingFee >= 0)
);
```

OFFLINE_BILL Table

a. Table definition

Field name	Data type	Default	Check	Key/Index/Constraint
BID	INTEGER			PK, FK, references to BILL(BID)
CustomerName	NVARCHAR(50)			
CustomerPhone	VARCHAR(15)			
EID	INTEGER			FK, references to EMPLOYEE(EID)
BrID	INTEGER			FK, references to BRANCH(BrID)

b. Example

BID	CustomerName	CustomerPhone	EID	BrID
2	Nguyen van a	123456789	1	1

c. Code

```
CREATE TABLE OFFLINE_BILL
(
  CustomerName NVARCHAR(50),
  CustomerPhone VARCHAR(15),
  EID INT NOT NULL,
  BrID INT NOT NULL,
  BID INT NOT NULL,
  PRIMARY KEY (BID),
  FOREIGN KEY (EID) REFERENCES EMPLOYEE(EID),
  FOREIGN KEY (BrID) REFERENCES BRANCH(BrID),
  FOREIGN KEY (BID) REFERENCES BILL(BID),

);
```

IN_BILL Table

a. Table definition

Field name	Data type	Default	Check	Key/Index/Constraint
PID	INTEGER			PK, FK, references to PRODUCT(PID)
BID	INTEGER			PK, FK, references to BILL(BID)
Amount	INTEGER		> 0	NOT NULL
DID	INTEGER			FK, references to DISCOUNT_PROGRAM(DID)

b. Example

PID	BID	Amount	DID
1	1	1	1
2	1	1	1
3	2	2	

4	2	2	

c. Code

```
CREATE TABLE IN_BILL
(
  Amount INT NOT NULL,
  BID INT NOT NULL,
  PID INT NOT NULL,
  DID INT NOT NULL,
  PRIMARY KEY (BID, PID),
  FOREIGN KEY (BID) REFERENCES BILL(BID),
  FOREIGN KEY (PID) REFERENCES PRODUCT(PID),
  FOREIGN KEY (DID) REFERENCES DISCOUNT_PROGRAM(DID),
  CONSTRAINT chk_ibamount CHECK (Amount > 0)
);
```

INVENTORY Table

a. Table definition

Field name	Data type	Default	Check	Key/Index/Constraint
PID	cINTEGER			PK, FK, references to PRODUCT(PID)
BrID	INTEGER			PK, FK, references to BRANCH(BrID)
Amount	INTEGER		>= 0	NOT NULL

b. Example

PID	BrID	Amount
1	1	100
2	1	200
3	2	300
4	3	500

c. Code

```
CREATE TABLE INVENTORY
(
  Amount INT NOT NULL,
  BrID INT NOT NULL,
  PID INT NOT NULL,
  PRIMARY KEY (BrID, PID),
  FOREIGN KEY (BrID) REFERENCES BRANCH(BrID),
  FOREIGN KEY (PID) REFERENCES PRODUCT(PID),
  CONSTRAINT chk_iamount CHECK (Amount >= 0)
);
```

LEADS Table

a. Table definition

Field name	Data type	Default	Check	Key/Index/Constraint
BrID	INTEGER			PK, FK, references to BRANCH(BrID)
EID	INTEGER			PK, FK, references to EMPLOYEE(EID)
StartedDate	DATETIME		<= GETDATE()	NOT NULL
EndedDate	DATETIME			

b. Example

BrID	EID	StartedDate	EndedDate
1	1	12/4/2021	
2	2	5/9/2021	5/10/2021

c. Code

```
CREATE TABLE LEADS
(
  StartedDate DATETIME NOT NULL,
  EndedDate DATETIME,
```

```
  BrID INT NOT NULL,
  EID INT NOT NULL,
  PRIMARY KEY (BrID, EID),
  FOREIGN KEY (BrID) REFERENCES BRANCH(BrID),
  FOREIGN KEY (EID) REFERENCES EMPLOYEE(EID),
  CONSTRAINT chk_leaddate CHECK (StartedDate <= GETDATE())
);
```

Adding Triggers/Stored Procedures

Triggers

Before inserting data into **BOOK Table**

```
CREATE TRIGGER TR_Book_Insert ON BOOK
INSTEAD OF INSERT
AS
BEGIN
      INSERT INTO BOOK
            SELECT b.* FROM inserted b
                  INNER JOIN PRODUCT p ON p.PID=b.PID
            WHERE p.PCategory='book' AND b.PubID <= p.PImportedDate;
END
```

Before inserting data into **STATIONERY Table**

```
CREATE TRIGGER TR_Stationery_Insert ON STATIONERY
INSTEAD OF INSERT
AS
BEGIN
      INSERT INTO STATIONERY
            SELECT s.* FROM inserted s
                  INNER JOIN PRODUCT p ON p.PID=s.PID
            WHERE p.PCategory='stationery';
END
```

Before inserting data into **ONLINE_BILL Table**

```
CREATE TRIGGER TR_OnBill_Insert ON ONLINE_BILL
INSTEAD OF INSERT
AS
BEGIN
      INSERT INTO ONLINE_BILL
```

```
            SELECT b.* FROM inserted b
                  INNER JOIN USERS u ON u.UID = b.UID
                  INNER JOIN BILL bb ON b.BID = bb.BID
            WHERE u.RegisterDate <= bb.BDate;
END
```

Before inserting data into **OFFLINE_BILL Table**

```
CREATE TRIGGER TR_OffBill_Insert ON OFFLINE_BILL
INSTEAD OF INSERT
AS
BEGIN
      INSERT INTO OFFLINE_BILL
            SELECT b.* FROM inserted b
                  INNER JOIN EMPLOYEE e ON e.EID = b.EID
                  INNER JOIN BILL bb ON b.BID = bb.BID
            WHERE e.InDate <= bb.BDate;
END
```

Before inserting data into **LEADS Table**

```
CREATE TRIGGER TR_Leads_Insert ON LEADS
INSTEAD OF INSERT
AS
BEGIN
      INSERT INTO LEADS
            SELECT l.* FROM inserted l
                  INNER JOIN EMPLOYEE e ON e.EID = l.EID
            WHERE e.InDate <= l.StartedDate;
END
```

Before inserting data into **VOUCHER Table**

```
CREATE TRIGGER TR_Voucher_Insert ON VOUCHER
INSTEAD OF INSERT
AS
BEGIN
      INSERT INTO VOUCHER
            SELECT v.* FROM inserted v
            WHERE v.StartedDate <= v.EndedDate;
END
```

Before inserting data into **DISCOUNT_PROGRAM Table**

```
CREATE TRIGGER TR_Discount_Insert ON DISCOUNT_PROGRAM
INSTEAD OF INSERT
AS
BEGIN
      INSERT INTO DISCOUNT_PROGRAM
            SELECT d.* FROM inserted d
            WHERE d.StartedDate <= d.EndedDate;
END
```

Before inserting data into **EMPLOYEE Table**

```
CREATE TRIGGER TR_Employee_Insert ON EMPLOYEE
INSTEAD OF INSERT
AS
BEGIN
      INSERT INTO EMPLOYEE
            SELECT e.* FROM inserted e
            WHERE e.InDate <= e.OutDate;
END
```

Stored Procedures

Update user rank due to purchase amount

We define:

- money < 100.000 VND: **newbie**
- 100.000 VND <= money <= 500.000 VND: **brown**
- 500.001 VND <= money <= 1.000.000 VND: **silver**
- 1.000.001 VND <= money <= 10.000.000 VND: **gold**
- money > 10.000.000 VND: **diamond**

```
CREATE PROCEDURE UpdateUserRank
AS
BEGIN
    WITH USERS_profit
    AS (SELECT u.UID,
               COALESCE(SUM(   pro.PPrice - (CASE
                                                 WHEN pro.PPrice *
d.DRate < d.DLimit THEN
                                                     pro.PPrice * d.DRate
                                                 ELSE
                                                     d.DLimit
```

```
                                                END
                                               ) * ib.Amount
                            ), 0) AS money_u
            FROM in_bill ib
                JOIN ONLINE_BILL on_b
                    ON ib.BID = on_b.BID
                JOIN PRODUCT pro
                    ON pro.PID = ib.PID
                JOIN DISCOUNT_PROGRAM d
                    ON d.DID = ib.DID
                RIGHT JOIN USERS u
                    ON on_b.UID = u.UID
            GROUP BY u.UID
           )
        UPDATE [USERS]
        SET [USERS].URank = CASE
                                WHEN USERS_profit.money_u < 100000
                                     OR USERS_profit.money_u IS NULL THEN
                                    'newbie'
                                WHEN (USERS_profit.money_u
                                     BETWEEN 100000 AND 500000
                                     ) THEN
                                    'brown'
                                WHEN (USERS_profit.money_u
                                     BETWEEN 500001 AND 1000000
                                     ) THEN
                                    'silver'
                                WHEN (USERS_profit.money_u
                                     BETWEEN 1000001 AND 10000000
                                     ) THEN
                                    'gold'
                                ELSE
                                    'diamond'
                            END
        FROM USERS_profit,
             USERS
        WHERE USERS_profit.UID = [USERS].UID;

    END
```

SQL Code for some request from company

TOP 100 Products which are Bestsellers

```
SELECT TOP 100
    pro.PID,
    pro.PName,
    sum(ib.Amount) AS ProductAmount
FROM
    IN_BILL     ib
    JOIN
        PRODUCT pro
            ON ib.PID = pro.PID
GROUP BY
    pro.PID,
    pro.PName
ORDER BY
    ProductAmount DESC;
```

TOP 100 Users which Buy the most

```
SELECT TOP 100
    u.UID,
    COALESCE(SUM(   pro.PPrice - (CASE
                                      WHEN pro.PPrice * d.DRate <
d.DLimit THEN
                                          pro.PPrice * d.DRate
                                      ELSE
                                          d.DLimit
                                  END
                                 ) * ib.Amount
                ), 0) AS Profit
FROM IN_BILL ib
    JOIN ONLINE_BILL on_b
        ON ib.BID = on_b.BID
    JOIN PRODUCT pro
        ON pro.PID = ib.PID
    JOIN DISCOUNT_PROGRAM d
        ON d.DID = ib.DID
    RIGHT JOIN USERS u
```

```
        ON u.UID = on_b.UID
GROUP BY u.UID
ORDER BY Profit DESC;
```

TOP 10 Authors which have Books in Bestsellers

```
SELECT TOP 10 au.AID,
           au.AName,
           au.ANationality,
           SUM(ib.Amount) AS BookAmount
FROM BOOK_AUTHOR au
JOIN BOOK bo ON au.AID=bo.AID
JOIN IN_BILL ib ON ib.PID=bo.PID
GROUP BY au.AID,
         au.AName,
         au.ANationality
ORDER BY BookAmount DESC;
```

Additional Link

- All file source: https://bit.ly/DBI202FinalProject
- Days of Week Codes: https://registrar.gmu.edu/topics/days-of-the-week-codes/