TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI TẬP LỚN MÔN HỆ CƠ SỞ DỮ LIỆU**

**QUẢN LÝ HỆ THỐNG ĐĂNG KÝ MÔN HỌC**

*Người hướng dẫn*:

*Người thực hiện*:

Lớp **:**

Khoá  **:**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI TẬP LỚN MÔN HỆ CƠ SỞ DỮ LIỆU**

**QUẢN LÝ HỆ THỐNG ĐĂNG KÝ MÔN HỌC**

Người hướng dẫn:

Người thực hiện:

Lớp **:**

Khoá  **:**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022**

# LỜI CẢM ƠN

Em xin gửi lời cảm ơn chân thành đến giảng viên bộ môn – … đã truyền đạt những kiến thức quý báu cho em trong suốt thời gian học tập vừa qua. Trong thời gian tham gia lớp học Hệ cơ sở dữ liệu của thầy, em đã có thêm cho mình nhiều kiến thức bổ ích qua đó thúc đẩy sự tò mò và hứng thú của em với môn học, ngành học hơn. Đây chắc chắn sẽ là những kiến thức quý báu, là hành trang để em có thể vững bước trong quá trình học tập của mình cũng như là khi đi làm.

Bài tập lớn khó có thể tránh khỏi những thiếu sót và nhiều chỗ còn chưa chính xác, kính mong Thầy xem xét và góp ý để bài tập lớn của em được hoàn thiện hơn.

Em xin chân thành cảm ơn.

*TP. Hồ Chí Minh, ngày tháng năm 2022*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

**BÀI TẬP LỚN ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Em xin cam đoan đây là bài tập lớn của riêng em và được sự hướng dẫn của …. Các nội dung trong bài tập lớn này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây.

**Nếu phát hiện có bất kỳ sự gian lận nào em xin hoàn toàn chịu trách nhiệm về nội dung bài tập lớn của mình.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do em gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày tháng năm 2022*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

**PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN**

**Phần xác nhận của GV hướng dẫn**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

**Phần đánh giá của GV chấm bài**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

# MỤC LỤC

[LỜI CẢM ƠN 1](#_Toc121088788)

[MỤC LỤC 4](#_Toc121088789)

[PHẦN 1 – HỆ THỐNG ĐĂNG KÝ MÔN HỌC 5](#_Toc121088790)

[1.1 Đặc tả 5](#_Toc121088791)

[1.2 Mô hình ERD 6](#_Toc121088792)

[1.3 Mô hình quan hệ 7](#_Toc121088793)

[1.4 Câu lệnh SQL 8](#_Toc121088794)

[*1.4.1 Tạo các bảng* 8](#_Toc121088795)

[*1.4.2 Function* 11](#_Toc121088796)

[*1.4.3 Trigger* 14](#_Toc121088797)

[PHẦN 2 16](#_Toc121088798)

[2.1 Câu 1 16](#_Toc121088799)

[2.2 Câu 2 26](#_Toc121088800)

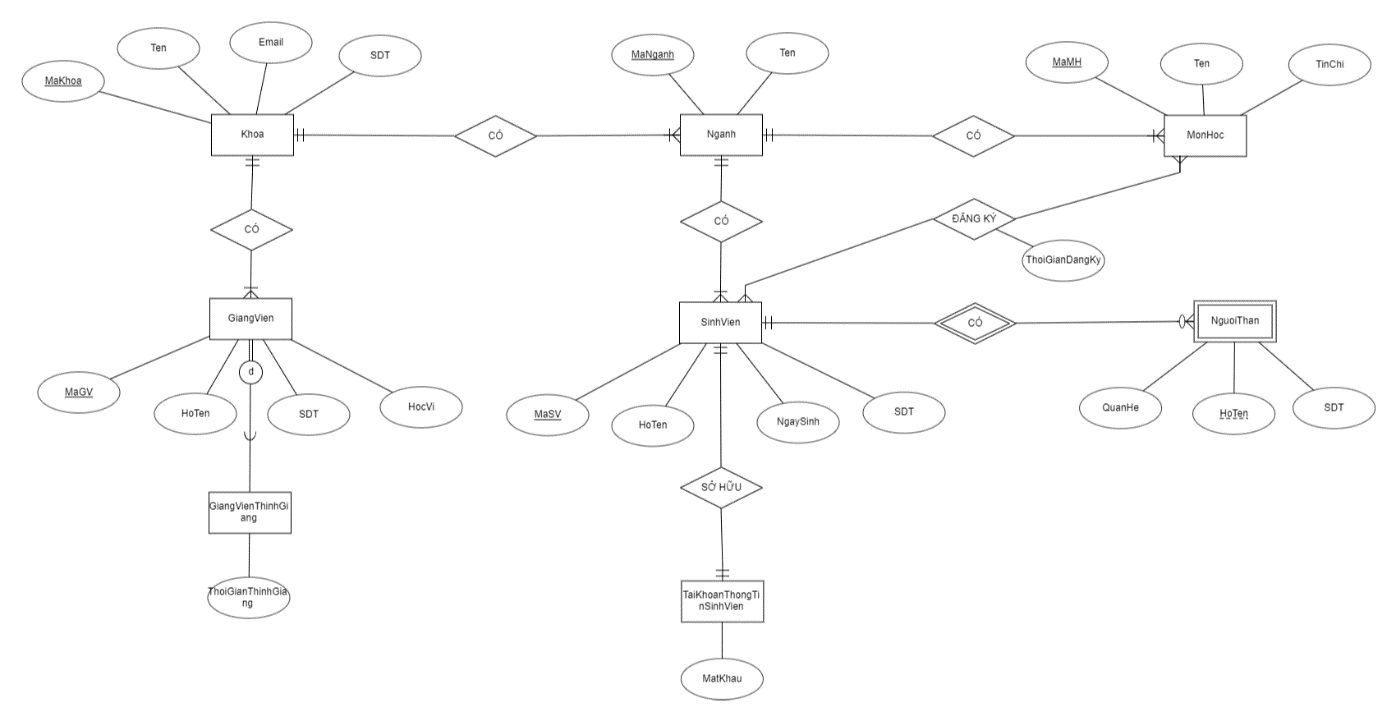
[TÀI LIỆU THAM KHẢO 33](#_Toc121088801)

# PHẦN 1 – HỆ THỐNG ĐĂNG KÝ MÔN HỌC

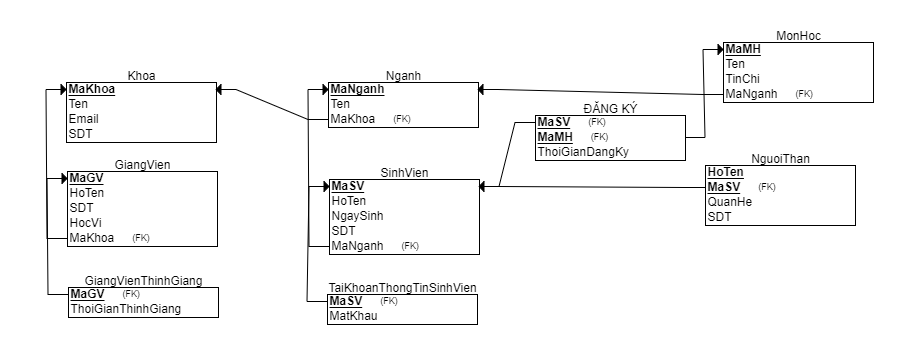
## 1.1 Đặc tả

* Một trường đại học cần xây dựng hệ thống đăng ký môn học.
* Một trường đại học gồm nhiều khoa. Thông tin của mỗi khoa bao gồm mã khoa, tên khoa, email khoa và số điện thoại khoa. Mỗi khoa gồm nhiều ngành. Thông tin về ngành gồm mã ngành, tên ngành. Mỗi ngành chỉ thuộc đúng một khoa.
* Mỗi khoa có nhiều giảng viên, tuy nhiên mỗi giảng viên chỉ thuộc một khoa duy nhất. Thông tin về giảng viên bao gồm mã giảng viên, họ tên giảng viên, số điện thoại, học vị. Bên cạnh giảng viên tại trường còn có những giảng viên thỉnh giảng. Đối với giảng viên thỉnh giảng cần lưu thêm thông tin như thời gian thỉnh giảng.
* Mỗi ngành có nhiều sinh viên. Thông tin về sinh viên bao gồm mã sinh viên, họ tên sinh viên, ngày sinh, số điện thoại. Mỗi sinh viên sẽ được nhà trường cung cấp một tài khoản thông tin sinh viên duy nhất. Thông tin về tài khoản thông tin sinh viên bao gồm mật khẩu.
* Mỗi ngành có nhiều môn học. Thông tin về môn học gồm mã môn học, tên môn học, số tín chỉ. Sinh viên được đăng ký nhiều môn học, mỗi môn học được đăng ký bởi nhiều sinh viên.
* Mỗi sinh viên có thể có nhiều người thân. Thông tin về người thân bao gồm quan hệ, họ tên, số điện thoại.

## 1.2 Mô hình ERD



## 1.3 Mô hình quan hệ



## 1.4 Câu lệnh SQL

### *1.4.1 Tạo các bảng*

CREATE TABLE Khoa

(

MaKhoa CHAR(10) NOT NULL,

Ten NVARCHAR(30) NOT NULL,

Email VARCHAR(50) NOT NULL,

SDT CHAR(10) NOT NULL,

PRIMARY KEY (MaKhoa)

);

CREATE TABLE Nganh

(

MaNganh CHAR(10) NOT NULL,

Ten NVARCHAR(30) NOT NULL,

MaKhoa CHAR(10) NOT NULL,

PRIMARY KEY (MaNganh),

FOREIGN KEY (MaKhoa) REFERENCES Khoa(MaKhoa)

);

CREATE TABLE GiangVien

(

MaGV CHAR(10) NOT NULL,

HoTen NVARCHAR(50) NOT NULL,

SDT CHAR(10) NOT NULL,

HocVi NVARCHAR(10) NOT NULL,

MaKhoa CHAR(10) NOT NULL,

PRIMARY KEY (MaGV),

FOREIGN KEY (MaKhoa) REFERENCES Khoa(MaKhoa)

);

CREATE TABLE GiangVienThinhGiang

(

MaGV CHAR(10) NOT NULL,

ThoiGianThinhGiang DATE NOT NULL,

PRIMARY KEY (MaGV),

FOREIGN KEY (MaGV) REFERENCES GiangVien(MaGV)

);

CREATE TABLE SinhVien

(

MaSV CHAR(10) NOT NULL,

HoTen NVARCHAR(50) NOT NULL,

NgaySinh DATE NOT NULL,

SDT CHAR(10) NOT NULL,

MaNganh CHAR(10) NOT NULL,

PRIMARY KEY (MaSV),

FOREIGN KEY (MaNganh) REFERENCES Nganh(MaNganh)

);

CREATE TABLE TaiKhoanThongTinSinhVien

(

MaSV CHAR(10) NOT NULL,

MatKhau VARCHAR(50) NOT NULL,

PRIMARY KEY (MaSV),

FOREIGN KEY (MaSV) REFERENCES SinhVien(MaSV)

);

CREATE TABLE NguoiThan

(

HoTen NVARCHAR(50) NOT NULL,

QuanHe NVARCHAR(10) NOT NULL,

SDT CHAR(10) NOT NULL,

MaSV CHAR(10) NOT NULL,

PRIMARY KEY (HoTen, MaSV),

FOREIGN KEY (MaSV) REFERENCES SinhVien(MaSV)

);

CREATE TABLE MonHoc

(

MaMH CHAR(10) NOT NULL,

Ten NVARCHAR(50) NOT NULL,

TinChi INT NOT NULL,

MaNganh CHAR(10) NOT NULL,

PRIMARY KEY (MaMH),

FOREIGN KEY (MaNganh) REFERENCES Nganh(MaNganh)

);

CREATE TABLE DangKy

(

ThoiGianDangKy DATE NOT NULL,

MaSV CHAR(10) NOT NULL,

MaMH CHAR(10) NOT NULL,

PRIMARY KEY (MaSV, MaMH),

FOREIGN KEY (MaSV) REFERENCES SinhVien(MaSV),

FOREIGN KEY (MaMH) REFERENCES MonHoc(MaMH)

);

### *1.4.2 Function*

* Function tạo mã sinh viên tự động (MaSV được tạo tự động theo quy tắc SVXXXXX, ví dụ: SV00001, SV00002,…)

CREATE FUNCTION generateMaSV()

RETURNS CHAR(10)

AS

BEGIN

IF (SELECT TOP 1 MaSV FROM SinhVien) IS NULL

RETURN 'SV00001'

DECLARE @STT INT

-- Vì kiểu dữ liệu trả về là CHAR(10) nhưng MaSV chỉ có 7 kí tự nên sẽ còn 3 kí tự khoảng trắng ở cuối

-- Vì vậy ta sẽ lấy 8 kí tự từ bên phải sang

SELECT TOP 1 @STT = CONVERT(INT, RIGHT(MaSV, 8)) FROM SinhVien ORDER BY MaSV DESC

SET @STT += 1

DECLARE @MaSV VARCHAR(10)

SET @MaSV = 'SV'

DECLARE @i INT

SET @i = LEN(CONVERT(VARCHAR(5), @STT))

WHILE @i < 5

BEGIN

SET @MaSV += '0'

SET @i += 1

END

SET @MaSV += CONVERT(VARCHAR(5), @STT)

RETURN RTRIM(@MaSV)

END

* Function tạo mã giảng viên tự động (MaGV được tạo tự động theo quy tắc GVXXXX, ví dụ: GV0001, GV0002,…)

CREATE FUNCTION generateMaGV()

RETURNS CHAR(10)

AS

BEGIN

IF (SELECT TOP 1 MaGV FROM GiangVien) IS NULL

RETURN 'GV0001'

DECLARE @STT INT

-- Vì kiểu dữ liệu trả về là CHAR(10) nhưng MaGV chỉ có 6 kí tự nên sẽ còn 4 kí tự khoảng trắng ở cuối

-- Vì vậy ta sẽ lấy 8 kí tự từ bên phải sang

SELECT TOP 1 @STT = CONVERT(INT, RIGHT(MaGV, 8)) FROM GiangVien ORDER BY MaGV DESC

SET @STT += 1

DECLARE @MaGV VARCHAR(10)

SET @MaGV = 'GV'

DECLARE @i INT

SET @i = LEN(CONVERT(VARCHAR(5), @STT))

WHILE @i < 4

BEGIN

SET @MaGV += '0'

SET @i += 1

END

SET @MaGV += CONVERT(VARCHAR(5), @STT)

RETURN RTRIM(@MaGV)

END

### *1.4.3 Trigger*

CREATE TRIGGER trigger\_ThemTaiKhoanThongTinSinhVien ON TaiKhoanThongTinSinhVien

AFTER INSERT

AS

BEGIN

DECLARE @MaSV CHAR(10)

DECLARE @MatKhau VARCHAR(50)

SELECT @MaSV = MaSV, @MatKhau = MatKhau FROM inserted

IF (@MaSV is NULL OR EXISTS (select @MaSV from SINHVIEN where MaSV = @MaSV)) -- kiểm tra ràng buộc khoá ngoại

BEGIN

IF ((SELECT COUNT(MaSV) FROM TaiKhoanThongTinSinhVien WHERE MaSV = @MaSV) = 2) -- kiểm tra tài khoản đã tồn tại hay chưa

BEGIN

PRINT(N'Sinh viên đã có tài khoản')

ROLLBACK TRAN

END

-- Kiểm tra mật khẩu

IF (@MatKhau IS NULL)

BEGIN

PRINT(N'Mật khẩu không được để trống')

ROLLBACK TRAN

END

IF (LEN(@MatKhau) < 8)

BEGIN

PRINT(N'Mật khẩu phải có nhiều hơn 8 ký tự')

ROLLBACK TRAN

END

END

ELSE

BEGIN

PRINT(N'Mã sinh viên không tồn tại')

ROLLBACK TRAN

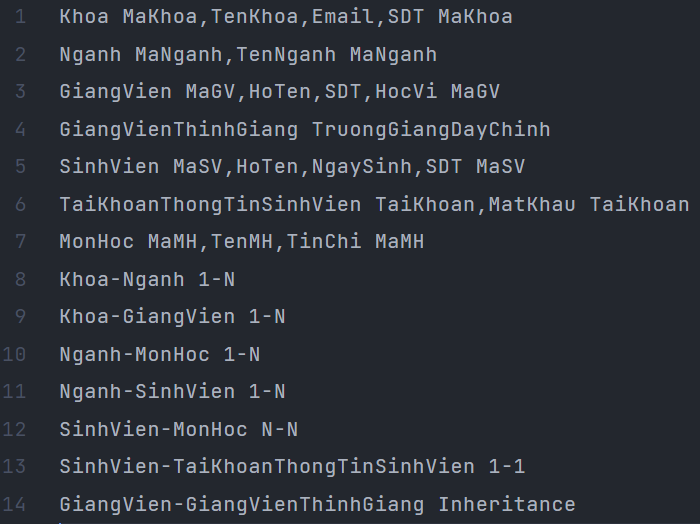
END

END

# PHẦN 2

## 2.1 Câu 1

* **Định dạng file input.txt**
  + Các thực thể của mô hình ERD: Tên thực thể <khoảng cách> Tập thuộc tính (các thuộc tính cách nhau bằng dấu phẩy) <khoảng cách> Khóa chính
  + Các mối quan hệ giữa các thực thể của mô hình ERD: Thực thể thứ nhất-Thực thể thứ hai <khoảng cách> Loại quan hệ
* **Ví dụ file input.txt**



* **Code**

import itertools

class ERD():

    def \_\_init\_\_(self, name, attributes, primaryKey):

        self.name = name

        self.attributes = attributes

        self.primaryKey = primaryKey

    def \_\_str\_\_(self):

        return self.name + "(" + self.attributes + ")" + " PK: " + self.primaryKey

class RelationalSchema():

    def \_\_init\_\_(self, name, attributes, primaryKey, foreignKey):

        self.name = name

        self.attributes = attributes

        self.primaryKey = primaryKey

        self.foreignKey = foreignKey

    def \_\_str\_\_(self):

        return self.name + "(" + self.attributes + ")" + " PK: " + self.primaryKey + " FK: " + self.foreignKey

class Relationship():

    def \_\_init\_\_(self, firstEntity, secondEntity, type):

        self.firstEntity = firstEntity

        self.secondEntity = secondEntity

        self.type = type

    def \_\_str\_\_(self):

        return self.firstEntity + "-" + self.secondEntity + ": " + self.type

def ERDToRelationalSchema(inputPath, outputPath):

    database = []  # Chứa các thực thể của mô hình ERD

    relationships = []  # Chứa mối quan hệ giữa các thực thể của mô hình ERD

    myInput = open(inputPath, 'r')

    lines = myInput.readlines()

    for line in lines:

        data = line.rstrip('\n').split(" ")

        if (len(data) == 3):

            database.append(ERD(data[0], data[1], data[2]))

        if (len(data) == 2):

            entities = data[0].split("-")

            relationships.append(Relationship(

                entities[0], entities[1], data[1]))

    resultDatabase = []  # Chứa các bảng dữ liệu

    resultRelationships = []  # Chứa mối quan hệ giữa các bảng

    # Mô hình quan hệ không có mối quan hệ N-N

    for relationship in relationships:

        if (relationship.type == 'N-N'):

            continue

        resultRelationships.append(relationship)

    existed = []  # Chứa tên các bảng đã tồn tại trong mảng resultDatabase, mục đích để không thêm trùng lặp các bảng

    for relationship in relationships:

        for firstEntity in database:

            if relationship.firstEntity == firstEntity.name:

                # Quan hệ 1-1

                if relationship.type == '1-1':

                    # Khoá chính ở phía bắt buộc làm khoá ngoại ở phía tuỳ chọn

                    foreignKey = firstEntity.primaryKey

                    for secondEntity in database:

                        if relationship.secondEntity == secondEntity.name:

                            if firstEntity.name not in existed:

                                resultDatabase.append(RelationalSchema(

                                    firstEntity.name, firstEntity.attributes, firstEntity.primaryKey, ''))

                                existed.append(firstEntity.name)

                            if secondEntity.name not in existed:

                                resultDatabase.append(RelationalSchema(

                                    secondEntity.name, secondEntity.attributes + ',' + foreignKey, secondEntity.primaryKey, foreignKey))

                                existed.append(secondEntity.name)

                # Quan hệ 1-N

                if relationship.type == '1-N':

                    # Khoá chính ở phía quan hệ một làm khoá ngoại ở phía quan hệ nhiều

                    foreignKey = firstEntity.primaryKey

                    for secondEntity in database:

                        if relationship.secondEntity == secondEntity.name:

                            if firstEntity.name not in existed:

                                resultDatabase.append(RelationalSchema(

                                    firstEntity.name, firstEntity.attributes, firstEntity.primaryKey, ''))

                                existed.append(firstEntity.name)

                            if secondEntity.name in existed:  # Trường hợp bảng có nhiều khóa ngoại

                                for entity in resultDatabase:

                                    if (entity.name == secondEntity.name):

                                        entity.attributes += (',' + foreignKey)

                                        entity.foreignKey += (',' + foreignKey)

                                        break

                            else:

                                resultDatabase.append(RelationalSchema(

                                    secondEntity.name, secondEntity.attributes + ',' + foreignKey, secondEntity.primaryKey, foreignKey))

                                existed.append(secondEntity.name)

                # Quan hệ N-N

                elif relationship.type == 'N-N':

                    for secondEntity in database:

                        if relationship.secondEntity == secondEntity.name:

                            print("Đây là mối quan hệ N-N giữa thực thể " +

                                  firstEntity.name + " và thực thể " + secondEntity.name)

                            nameNewTable = input(

                                "Vui lòng nhập tên quan hệ mới được tạo ra từ quan hệ N-N trên: ")

                            attributesNewTable = input(

                                "Vui lòng nhập các thuộc tính muốn thêm cho quan hệ này: ")

                            resultDatabase.append(RelationalSchema(nameNewTable, firstEntity.primaryKey + ',' + secondEntity.primaryKey + ',' + attributesNewTable,

                                                                   firstEntity.primaryKey + ',' + secondEntity.primaryKey, firstEntity.primaryKey + ',' + secondEntity.primaryKey))

                            existed.append(nameNewTable)

                            resultRelationships.append(Relationship(

                                firstEntity.name, nameNewTable, "1-N"))

                            resultRelationships.append(Relationship(

                                secondEntity.name, nameNewTable, "1-N"))

                # Quan hệ kế thừa

                elif relationship.type == 'Inheritance':

                    # Khoá chính của quan hệ cha trở thành khoá chính của quan hệ con

                    primaryKey = firstEntity.primaryKey

                    for secondEntity in database:

                        if relationship.secondEntity == secondEntity.name:

                            if firstEntity.name not in existed:

                                resultDatabase.append(RelationalSchema(

                                    firstEntity.name, firstEntity.attributes, firstEntity.primaryKey, ''))

                                existed.append(firstEntity.name)

                            if secondEntity.name not in existed:

                                resultDatabase.append(RelationalSchema(

                                    secondEntity.name, secondEntity.attributes + ',' + primaryKey, primaryKey, primaryKey))

                                existed.append(secondEntity.name)

    myOutput = open(outputPath, 'w')

    for i in range(len(resultDatabase)):

        myOutput.writelines(resultDatabase[i].\_\_str\_\_())

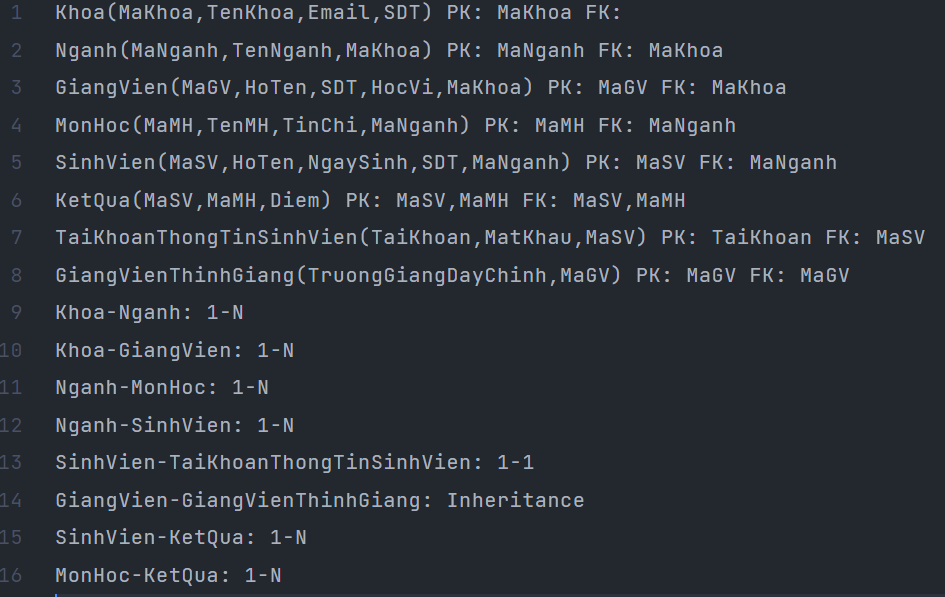
        myOutput.write("\n")

    for relationship in resultRelationships:

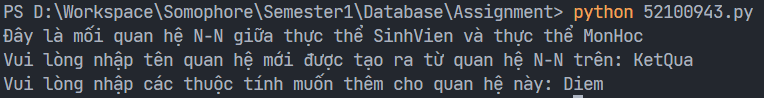
        myOutput.writelines(relationship.\_\_str\_\_())

        myOutput.write("\n")

* **Định dạng file output.txt**
  + Các bảng dữ liệu: Tên bảng(Các thuộc tính ngăn cách nhau bằng dấu phẩy) <khoảng cách> PK: Khóa chính <khoảng cách> FK: Khóa ngoại
  + Các mối quan hệ giữa các bảng: Thực thể thứ nhất-Thực thể thứ hai: <khoảng cách> Loại quan hệ
* **Ví dụ file output.txt**

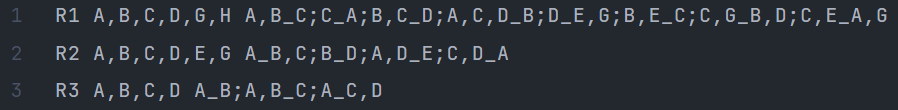


* **Lưu** **ý**: Đối với mối quan hệ N-N của mô hình ERD khi chuyển sang mô hình quan hệ sẽ tạo ra quan hệ mới. Lúc này chương trình sẽ yêu cầu người dùng nhập tên quan hệ mới và các thuộc tính muốn thêm cho quan hệ mới
  + Ví dụ



## 2.2 Câu 2

* **Định dạnh file input.txt**
  + Mỗi dòng bao gồm tên bảng <khoảng cách> Các thuộc tính của bảng được ngăn cách bằng dấu phẩy <khoảng cách> Các phụ thuộc hàm được ngăn cách bằng dấu chấm phẩy (dấu \_ để ngăn cách vế trái phụ thuộc hàm và vế phải phụ thuộc hàm, dấu phẩy ngăn cách các thuộc tính trong một vế của phụ thuộc hàm
* **Ví dụ file input.txt**



* **Code**

def convertFDSetToString(elements):

    return "{{{}}}".format(", ".join(sorted([str(element) for element in elements])))

def convertSetToString(set):

    return "".join(attribute for attribute in set)

def convertStringToSet(str):

    attributes = str.split(",")

    result = set()

    for attribute in attributes:

        result.add(attribute)

    return result

def sortString(str):

    return ''.join(sorted(str))

class FunctionalDependency:

    def \_\_init\_\_(self, lhs, rhs):

        self.lhs = lhs

        self.rhs = rhs

    def \_\_str\_\_(self):

        return "{} -> {}".format(convertSetToString(self.lhs), convertSetToString(self.rhs))

class FDSet(set):

    def \_\_init\_\_(self):

        set.\_\_init\_\_(self)

    def \_\_str\_\_(self):

        return convertFDSetToString(self)

    def setOfAttributesInLeftHandSide(self):

        result = set()

        for fd in self:

            result.update(fd.lhs)

        return result

    def setOfAttributesInRightHandSide(self):

        result = set()

        for fd in self:

            result.update(fd.rhs)

        return result

    # Tìm bao đóng của tập thuộc tính

    def closureOfAttributeSet(self, attributes):

        result = set(attributes)

        stillChange = True

        while stillChange:

            stillChange = False

            for fd in self:

                if fd.lhs.issubset(result) and not fd.rhs.issubset(result):

                    stillChange = True

                    result.update(fd.rhs)

        return result

    # Tìm các khóa

    def keys(self, attributes):

        N = attributes.difference(self.setOfAttributesInRightHandSide())

        if convertSetToString(self.closureOfAttributeSet(N)) == convertSetToString(attributes):

            return convertSetToString(N)

        D = self.setOfAttributesInRightHandSide().difference(

            self.setOfAttributesInLeftHandSide())

        L = attributes.difference(N.union(D))

        powerSet = []

        for i in range(len(L)):

            for element in list(itertools.combinations(L, i + 1)):

                powerSet.append(element)

        result = []

        listLiIsKey = []

        for Li in powerSet:

            isKey = False

            Li = set(Li)

            for element in listLiIsKey:

                if element.issubset(Li):

                    isKey = True

            if (isKey):

                continue

            X = N.union(Li)

            if sortString(convertSetToString(self.closureOfAttributeSet(convertSetToString(X)))) == sortString(convertSetToString(attributes)):

                result.append(X)

                listLiIsKey.append(Li)

        for i in range(len(result)):

            result[i] = convertSetToString(result[i])

        return result

class Relation():

    def \_\_init\_\_(self):

        self.name = ""

        self.attributes = ""

        self.fdSet = FDSet()

def countLinesInFile(filePath):

    count = 0

    myFile = open(filePath, "r")

    lines = myFile.readlines()

    for line in lines:

        count += 1

    return count

def loadFile(filePath):

    relations = []

    for i in range(countLinesInFile(filePath)):

        relations.append(Relation())

    myInput = open(filePath, "r")

    lines = myInput.readlines()

    i = 0

    for line in lines:

        data = line.rstrip('\n').split(" ")

        relations[i].name = data[0]

        relations[i].attributes = data[1]

        fdSet = data[2].split(";")

        for fd in fdSet:

            relations[i].fdSet.add(FunctionalDependency(convertStringToSet(fd.split(

                "\_")[0]), convertStringToSet(fd.split("\_")[1])))

        i += 1

    return relations

def writeToFile(filePath, relations, X):

    myOutput = open(filePath, "w")

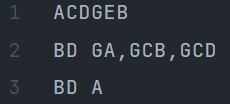
    for relation in relations:

        myOutput.writelines(convertSetToString(relation.fdSet.closureOfAttributeSet(X)) + " " + ','.join(relation.fdSet.keys(

            convertStringToSet(relation.attributes))))

        myOutput.write("\n")

* **Định dạnh file output.txt:** Bao đóng của tập thuộc tính X <khoảng cách> Các khóa của lược đồ quan hệ
* **Ví dụ file output.txt**



* **Lưu** **ý**: Chương trình sẽ yêu cầu người dùng nhập tập thuộc tính X
  + Ví dụ



# TÀI LIỆU THAM KHẢO

**Tiếng Việt**

**Tiếng Anh**

[1]. [functional\_dependencies API documentation (oer.gitlab.io)](https://oer.gitlab.io/cs/functional-dependencies/functional_dependencies.html#functional_dependencies.FDSet.closure)