



# CƠ SỞ DỮ LIỆU

## Giáo Trình SQL Cơ Bản

Tác giả: Trần Hoài Thuận

Chuyên ngành: Khai phá dữ liệu

Ngày: 30 Tháng 01, 2023

Version: 1.0



*Victory won't come to us unless we go to it.*

# Mục lục

<b>Chương 1 TỔNG QUAN VỀ CƠ SỞ DỮ LIỆU</b>	<b>1</b>
1.1 Dữ liệu . . . . .	1
1.2 Cơ sở dữ liệu . . . . .	2
1.3 Truy vấn dữ liệu . . . . .	4
1.4 Hệ quản trị CSDL . . . . .	4
1.5 Hệ CSDL . . . . .	5
1.6 Các chức năng của một hệ quản trị CSDL . . . . .	5
1.7 Quá trình phát triển của hệ CSDL . . . . .	6
1.8 Vai trò của con người trong hệ CSDL . . . . .	8
1.9 Các thành phần của hệ quản trị CSDL . . . . .	10
1.10 Mô hình dữ liệu . . . . .	11
1.11 Một vài ứng dụng cơ sở dữ liệu . . . . .	16
CHƯƠNG 1. Bài tập . . . . .	17
<b>Chương 2 MÔ HÌNH THỰC THỂ KẾT HỢP</b>	<b>18</b>
2.1 Thực thể và tập thực thể . . . . .	18
2.2 Thuộc tính . . . . .	18
2.3 Mối kết hợp và tập mối kết hợp . . . . .	20
2.4 Ví dụ mô hình thực thể kết hợp . . . . .	23
2.5 Các loại ký hiệu trong mô hình ERD . . . . .	24
CHƯƠNG 2. Bài tập . . . . .	27
<b>Chương 3 MÔ HÌNH DỮ LIỆU QUAN HỆ</b>	<b>38</b>
3.1 Các khái niệm của mô hình dữ liệu quan hệ . . . . .	38
3.2 Một số tính chất của quan hệ . . . . .	40
3.3 Các khái niệm về khóa . . . . .	41
3.4 Chuyển mô hình thực thể kết hợp sang mô hình quan hệ . . . . .	43
3.5 Ví dụ mô hình dữ liệu quan hệ . . . . .	46
CHƯƠNG 3. Bài tập . . . . .	47
<b>Chương 4 NGÔN NGỮ TRUY VẤN DỮ LIỆU</b>	<b>51</b>
4.1 Giới thiệu . . . . .	51
4.2 RDBMS . . . . .	51
4.3 Giá trị NULL . . . . .	53
4.4 Toàn vẹn dữ liệu . . . . .	53
4.5 Ràng buộc dữ liệu . . . . .	53
4.6 Các kiểu dữ liệu trong SQL . . . . .	54

<b>Chương 5 CÁC LỆNH SQL CƠ BẢN</b>	<b>57</b>
5.1 Chú thích trong SQL . . . . .	57
5.2 Tạo Database . . . . .	58
5.3 Tạo Bảng . . . . .	59
5.4 Câu lệnh INSERT INTO . . . . .	60
5.5 Câu lệnh ALTER . . . . .	61
5.6 Câu lệnh DROP . . . . .	62
5.7 Câu lệnh TRUNCATE . . . . .	62
5.8 Câu lệnh SELECT . . . . .	63
5.9 Mệnh đề WHERE . . . . .	64
5.10 Toán tử AND, OR, NOT . . . . .	65
5.11 Toán tử IN . . . . .	66
5.12 Toán tử EXISTS . . . . .	67
5.13 Toán tử BETWEEN . . . . .	68
5.14 Toán tử UNION . . . . .	69
5.15 Câu lệnh ANY, ALL . . . . .	70
5.16 Câu lệnh AS . . . . .	71
5.17 Câu lệnh WITH . . . . .	72
5.18 Câu lệnh CASE . . . . .	73
5.19 Câu lệnh UPDATE . . . . .	74
5.20 Câu lệnh DELETE . . . . .	75
5.21 Mệnh đề LIKE . . . . .	76
5.22 Mệnh đề ORDER BY . . . . .	77
5.23 Mệnh đề LIMIT . . . . .	78
5.24 Mệnh đề GROUP BY . . . . .	79
5.25 Mệnh đề HAVING . . . . .	80
5.26 Mệnh đề JOIN . . . . .	81
5.27 Câu lệnh INNER JOIN . . . . .	82
5.28 Câu lệnh LEFT JOIN . . . . .	83
5.29 Câu lệnh RIGHT JOIN . . . . .	85
5.30 Câu truy vấn con . . . . .	87
<b>Chương 6 CÁC LOẠI RÀNG BUỘC</b>	<b>88</b>
6.1 Ràng buộc NOT NULL . . . . .	89
6.2 Ràng buộc UNIQUE . . . . .	90
6.3 Ràng buộc PRIMARY KEY . . . . .	92
6.4 Ràng buộc FOREIGN KEY . . . . .	94
6.5 Ràng buộc CHECK . . . . .	96
6.6 Ràng buộc DEFAULT . . . . .	97
6.7 Câu lệnh INDEX . . . . .	98

6.8 AUTO INCREMENT . . . . .	99
<b>Chương 7 CÁC HÀM CƠ BẢN TRONG MYSQL</b>	<b>101</b>
7.1 Hàm SUM . . . . .	103
7.2 Hàm MAX . . . . .	104
7.3 Hàm MIN . . . . .	105
7.4 Hàm AVG . . . . .	106
7.5 Hàm COUNT . . . . .	107
7.6 Hàm LENGTH, CHAR LENGTH . . . . .	108
7.7 Hàm CONCAT, CONCAT WS . . . . .	108
7.8 Hàm LOWER, UPPER . . . . .	109
7.9 Hàm FORMAT . . . . .	109
7.10 Hàm REPLACE . . . . .	109
7.11 Hàm COALESCE . . . . .	109
7.12 Hàm SUBSTRING, SUBSTR . . . . .	110
7.13 Hàm STRCMP . . . . .	111
7.14 Hàm SPACE . . . . .	111
7.15 Hàm LEFT, RIGHT . . . . .	111
7.16 Hàm TRIM, LTRIM, RTRIM . . . . .	112
7.17 Hàm INSERT . . . . .	113
7.18 Hàm INSTR . . . . .	113
7.19 Hàm FIELD . . . . .	114
7.20 Hàm FIND IN SET . . . . .	114
7.21 Hàm IFNULL, NULLIF . . . . .	115
7.22 Hàm ISNULL . . . . .	116
7.23 Hàm IF . . . . .	116
<b>Chương 8 SQL NÂNG CAO</b>	<b>118</b>
8.1 Function và Store Procedure . . . . .	118
8.2 Trigger . . . . .	123
8.3 View . . . . .	124
8.4 Transaction . . . . .	125
8.5 Cursor . . . . .	128
<b>Chương 9 BÀI TẬP SQL</b>	<b>130</b>

# CHƯƠNG 1. TỔNG QUAN VỀ CƠ SỞ DỮ LIỆU

Lưu trữ thông tin là nhu cầu rất thông dụng trong cuộc sống của con người. Từ thời xa xưa con người đã biết sử dụng các vật dụng có sẵn để lưu trữ thông tin như in hình lên đá, in lên gốc cây, thanh gỗ,... Nhưng với sự phát triển vượt bậc của công nghệ thông tin, lượng thông tin cần lưu trữ là rất lớn và con người lưu trữ những thông tin này trên các thiết bị điện tử ví dụ như máy tính, smartphone,... Để nguồn thông tin được lưu trữ có tính đồng nhất thì việc sử dụng hệ quản trị cơ sở dữ liệu là cần thiết.

Cơ sở dữ liệu (CSDL) và các hệ thống CSDL trở thành một phần quan trọng trong cuộc sống của chúng ta ngày nay. Hầu hết các hoạt động hàng ngày đều gắn liền với sự giao tiếp với một CSDL, ví dụ như đến ngân hàng để rút tiền và gửi tiền, đăng ký chỗ trên máy bay hoặc khách sạn, truy cập vào thư viện để tin học hoá để tìm sách báo, đặt mua tạp chí ở một nhà xuất bản... tất cả các hoạt động này đều thông qua một chương trình máy tính để truy cập đến CSDL.

Tất cả các giao tiếp như trên được gọi là các ứng dụng của CSDL truyền thống. Trong các CSDL truyền thống, hầu hết các thông tin được lưu giữ và truy cập là văn bản hoặc số. Những năm gần đây, những tiến bộ về kỹ thuật đã đưa đến những ứng dụng mới của CSDL. Các *CSDL đa phương tiện* (*Multimedia Databases*) có thể lưu trữ hình ảnh, video và âm thanh. Các *hệ thống thông tin địa lý* (*Geographic Information System - GIS*) có thể lưu trữ và phân tích các bản đồ, các dữ liệu về thời tiết và các ảnh vệ tinh. *Kho dữ liệu* (*Data Warehouse*) và các *hệ thống phân tích trực tuyến* (*Online Analytical Processing - OLAP*) được sử dụng trong nhiều công ty để lấy ra và phân tích những thông tin có lợi từ các CSDL lớn nhằm đưa ra các quyết định. Kỹ thuật *CSDL động và thời gian thực* (*Real Time and Active Database Technology*) được sử dụng trong việc kiểm tra các tiến trình công nghiệp và sản xuất. Các kỹ thuật nghiên cứu CSDL được áp dụng thông qua các ứng dụng Web hoặc các ứng dụng Winform để người dùng khai thác, tìm kiếm thông tin cần thiết thông qua chương trình ứng dụng.

Để hiểu được nền tảng của các kỹ thuật CSDL, trước tiên chúng ta sẽ bắt đầu từ các cơ sở kỹ thuật của CSDL truyền thống. Trong chương này chúng ta sẽ tìm hiểu về các định nghĩa dữ liệu, CSDL, hệ quản trị CSDL, mô hình CSDL và các thuật ngữ cơ bản khác.

## 1.1 Dữ liệu

**Dữ liệu (Data)** là toàn bộ những thông tin dưới dạng ký hiệu chữ viết, con số, hình ảnh, âm thanh,... được máy tính lưu trữ và xử lý. Dữ liệu được phân làm 2 loại: *dữ liệu tĩnh* và *dữ liệu động*. Ví dụ: tên, địa chỉ, số điện thoại của khách hàng là những dữ liệu tĩnh; báo cáo doanh thu, đăng ký học phần là những dữ liệu động.

**Thông tin** là khái niệm phản ánh tri thức, sự hiểu biết của con người về một sự vật, hiện tượng, thực thể hay khái niệm nào đó mà ta có thể thu thập, lưu trữ và xử lý được chúng.

**Phân biệt giữa dữ liệu và thông tin:** ta có thể ví dụ dữ liệu như là nguyên vật liệu được dùng trong một quá trình sản xuất sản phẩm, còn thông tin như là sản phẩm. Vậy dữ liệu là một khái niệm rộng, thô, rời rạc và thông tin là sản phẩm của quá trình kết xuất từ dữ liệu. Nói đơn giản, thông tin chính là nội dung của dữ liệu sau khi đã được xử lý.

## 1.2 Cơ sở dữ liệu

**Cơ sở dữ liệu (Database)** là một tập hợp các dữ liệu có mối liên hệ chặt chẽ với nhau, được lưu trữ trên máy tính, có nhiều người sử dụng và được tổ chức theo một mô hình. Dữ liệu ở đây được hiểu là những gì *có thể ghi nhận lại và có ý nghĩa rõ ràng*. Ví dụ, xem xét tên, số điện thoại và địa chỉ của một người, ta có thể ghi nhận lại dữ liệu này trong một cuốn sổ tay, hoặc lưu trữ trong máy tính cá nhân có phần mềm hỗ trợ như Microsoft Access hoặc Excel. Đây là một tập hợp dữ liệu có liên quan với nhau và có ý nghĩa rõ ràng nên nó được xem là một CSDL.

Một CSDL thường bao gồm một hoặc nhiều bảng (table). Mỗi bảng được xác định thông qua một tên (ví dụ như Sinh viên). Bảng chứa các cột (column), dòng (record - row) là dữ liệu của bảng.

Chúng ta hãy xem xét một ví dụ quen thuộc sau:

Một CSDL lưu trữ thông tin liên quan đến sinh viên, ngành học, môn thi và kết quả thi của sinh viên trong một trường đại học. CSDL được tổ chức trong 5 bảng, mỗi bảng lưu trữ các bộ dữ liệu cùng loại. Bảng SINHVIEN lưu trữ dữ liệu cho mỗi sinh viên, bảng NGANH lưu trữ dữ liệu cho các ngành học, bảng MONTHI lưu trữ dữ liệu cho các môn thi, bảng NG\_MT lưu trữ thông tin môn thi của từng ngành, bảng KETQUA lưu trữ điểm sinh viên đạt được cho mỗi môn học.

SINHVIEN			
MSSV	HOTEN	GIOITINH	MANGANH
1510410001	Nguyễn Thị Lan Anh	Nữ	KTPM
1510410002	Trần Thị Kim Hoàn	Nữ	KTPM
1510410003	Vũ Minh Quân	Nam	HTTT

MONTHI		
MAMT	TENMT	GHICHU
TT1501	Kinh tế Chính trị Mác-Lênin	Môn chung
BD1502	Dữ liệu lớn	Môn cơ sở ngành
KT1503	Kỹ thuật lập trình	Môn cơ sở ngành

NGANH	
MANGANH	TENNGANH
KTPM	Kỹ thuật phần mềm
HTTT	Hệ thống thông tin

NG_MT	
MANGANH	MAMT
KTPM	TT1501
KTPM	KT1503
HTTT	TT1501
HTTT	BD1502

KETQUA		
MASV	MAMT	DIEMTHI
1510410001	TT1501	8.5
1510410001	KT1503	8.0
1510410002	TT1501	10.0
1510410002	KT1503	7.0
1510410003	TT1501	8.5
1510410003	BD1502	9.0

### Một CSDL có các tính chất sau:

1. Một CSDL phải biểu thị một khía cạnh nào đó của thế giới thực như hoạt động của một công ty, một nhà trường, một ngân hàng... Những thay đổi của thế giới thực phải được phản ánh một cách trung thực vào trong CSDL. Những thông tin được đưa vào trong CSDL tạo thành một không gian CSDL hoặc là một “thế giới nhỏ” (miniworld).
2. Một CSDL là một tập hợp dữ liệu liên kết với nhau một cách logic và mang một ý nghĩa cố hữu nào đó. Một CSDL không phải là một tập hợp tùy tiện.
3. Một CSDL được thiết kế và được phổ biến cho một mục đích riêng. Nó có một nhóm người sử dụng có chủ định và có một số ứng dụng được xác định phù hợp với mối quan tâm của người sử dụng. Nói cách khác, một CSDL có một nguồn cung cấp dữ liệu, một mức độ tương tác với các sự kiện trong thế giới thực và một nhóm người quan tâm tích cực đến các nội dung của nó.

Một CSDL có thể có cỡ tuỳ ý và có độ phức tạp thay đổi. Có những CSDL chỉ gồm vài trăm bản ghi ví dụ như CSDL phục vụ việc quản lý lương ở một cơ quan nhỏ, và có những CSDL có dung lượng rất lớn ví dụ như các CSDL phục vụ cho việc tính cước điện thoại, quản lý nhân sự trên một phạm vi lớn. Các CSDL phải được tổ chức quản lý sao cho những người sử dụng có thể tìm kiếm dữ liệu, cập nhật dữ liệu và lấy dữ liệu ra khi cần thiết. Một CSDL có thể được tạo ra và duy trì một cách thủ công và cũng có thể được tin học hóa. Một CSDL tin học hóa được tạo ra và duy trì bằng một nhóm chương trình ứng dụng hoặc bằng một *hệ quản trị cơ sở dữ liệu*.

## 1.3 Truy vấn dữ liệu

**Truy vấn dữ liệu** là việc thu thập các nguồn dữ liệu từ một CSDL và định dạng nó trong một biểu mẫu có thể đọc được. Truy vấn phải được viết bằng ngôn ngữ truy vấn dữ liệu (ví dụ như SQL). Hiểu theo cách đơn giản thì truy vấn dữ liệu là sử dụng bộ lọc để thu thập các thông tin.

Truy vấn dữ liệu trích xuất thông tin đang lưu trữ trong bảng. Thông tin được truy xuất thông qua cột và thông tin cần trích xuất có thể thuộc một hay nhiều bảng. Thao tác này được sử dụng phổ biến trong các hệ thống phần mềm hoặc website. Ví dụ khi bạn đăng nhập vào Zalo thì hệ thống sẽ thực hiện truy vấn dữ liệu để kiểm tra tích hợp điều lệ tài khoản đang đăng nhập,...

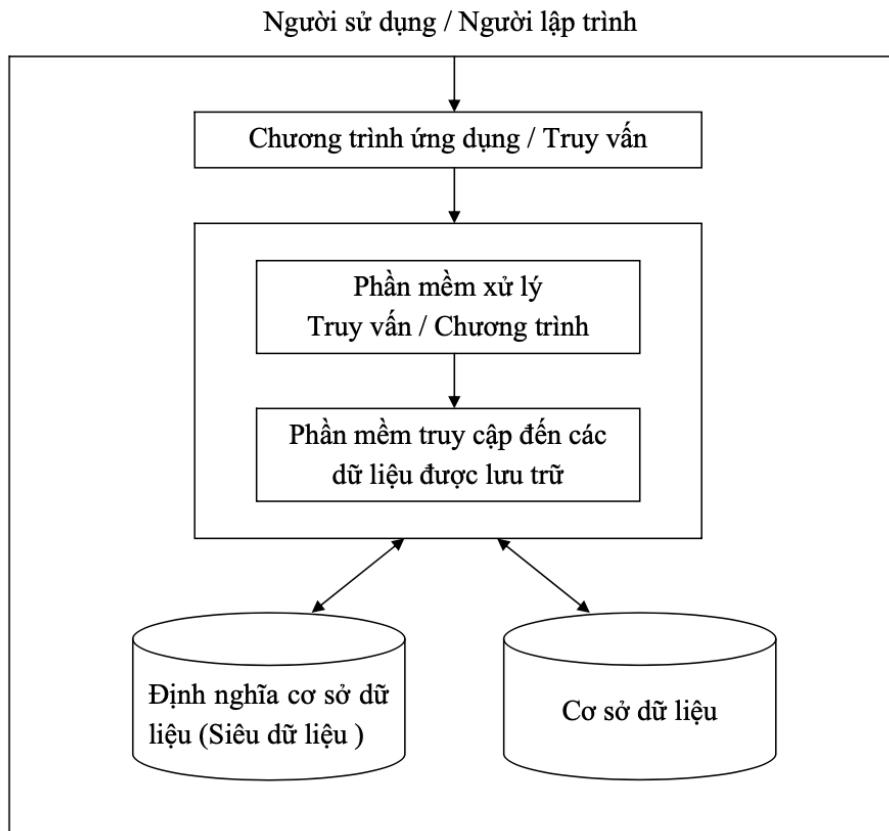
## 1.4 Hệ quản trị CSDL

**Hệ quản trị cơ sở dữ liệu (Database Management System – DBMS)** là tập hợp tất cả các chương trình cho phép người dùng tạo ra, duy trì và khai thác một cơ sở dữ liệu. Nó là một hệ thống phần mềm dùng để làm dễ quá trình *định nghĩa, xây dựng, xử lý và chia sẻ* CSDL giữa những người dùng khác nhau và ứng dụng.

1. *Định nghĩa CSDL*: gắn liền với việc chỉ định kiểu dữ liệu, cấu trúc và ràng buộc của dữ liệu được lưu trữ trong CSDL.
2. *Xây dựng CSDL*: là việc lưu trữ dữ liệu trên bộ nhớ phụ được điều khiển bởi một hệ quản trị cơ sở dữ liệu.
3. *Xử lý CSDL*: bao gồm các thao tác như truy xuất dữ liệu theo yêu cầu.
4. *Chia sẻ CSDL*: cho phép nhiều người dùng và nhiều ứng dụng cùng truy xuất đến CSDL. Ngoài ra, hệ quản trị CSDL còn cung cấp một giao diện để giao tiếp giữa người sử dụng và CSDL.

## 1.5 Hệ CSDL

Người ta gọi CSDL và hệ quản trị CSDL bằng một thuật ngữ chung là **hệ CSDL**. Môi trường của một hệ CSDL được mô tả bằng hình vẽ dưới đây:



**Hình 1.1:** Minh họa hệ CSDL

## 1.6 Các chức năng của một hệ quản trị CSDL

Một hệ quản trị CSDL hiện nay có các chức năng sau:

1. Lưu trữ các định nghĩa, các mối liên kết dữ liệu (gọi là *siêu dữ liệu*) vào một từ điển dữ liệu. Các chương trình truy cập đến cơ sở dữ liệu làm việc thông qua hệ quản trị cơ sở dữ liệu. Hệ quản trị cơ sở dữ liệu sử dụng dữ liệu trong từ điển dữ liệu để tìm kiếm các cấu trúc thành phần dữ liệu và các mối liên kết được yêu cầu. Mọi sự thay đổi trong các tệp cơ sở dữ liệu sẽ được tự động ghi lại vào từ điển dữ liệu. Như vậy, hệ quản trị cơ sở dữ liệu giải phóng người sử dụng khỏi việc lập trình cho các mối liên kết phức tạp trong mỗi chương trình, việc sửa đổi các chương trình truy cập đến tệp cơ sở dữ liệu đã bị sửa đổi. Nói cách khác, hệ quản trị cơ sở dữ liệu loại bỏ sự phụ thuộc giữa dữ liệu và cấu trúc ra khỏi hệ thống.
2. Tạo ra các cấu trúc phức tạp theo yêu cầu để lưu trữ dữ liệu. Nó giúp người sử dụng làm nhiệm vụ khó khăn là định nghĩa và lập trình cho các đặc trưng vật lý của dữ liệu.

3. Biến đổi các dữ liệu được nhập vào để phù hợp với các cấu trúc dữ liệu ở điểm 2. Như vậy, hệ quản trị cơ sở dữ liệu giúp người sử dụng phân biệt dạng logic và dạng vật lý của dữ liệu. Bằng việc duy trì sự độc lập dữ liệu, hệ quản trị cơ sở dữ liệu chuyển các yêu cầu logic thành các lệnh định vị một cách vật lý và lấy ra các dữ liệu yêu cầu. Điều đó cũng có nghĩa là hệ quản trị cơ sở dữ liệu tạo khuôn dạn cho các dữ liệu được lấy ra để làm cho nó phù hợp với mong muốn logic của người sử dụng.
4. Tạo ra một hệ thống bảo mật và áp đặt tính bảo mật và riêng tư trong cơ sở dữ liệu.
5. Tạo ra các cấu trúc phức tạp cho phép nhiều người sử dụng truy cập đến dữ liệu.
6. Cung cấp các thủ tục sao lưu và phục hồi dữ liệu để đảm bảo sự an toàn và toàn vẹn dữ liệu.
7. Xúc tiến và áp đặt các quy tắc an toàn để loại bỏ vấn đề toàn vẹn dữ liệu. Điều đó cho phép ta làm tối thiểu sự dư thừa dữ liệu và làm tối đa tính nhất quán dữ liệu.
8. Cung cấp việc truy cập dữ liệu thông qua một ngôn ngữ truy vấn. Một ngôn ngữ truy vấn là một ngôn ngữ phi thủ tục cho phép người sử dụng chỉ ra cái gì cần phải làm mà không cần phải chỉ ra nó được làm như thế nào. Các hệ quản trị cơ sở dữ liệu cũng cung cấp việc truy cập dữ liệu cho những người lập trình thông qua các ngôn ngữ thủ tục.

## 1.7 Quá trình phát triển của hệ CSDL

### 1.7.1 Lưu trữ dữ liệu dưới dạng hệ thống tập tin

Trước đây người ta thường sử dụng hệ thống tập tin để lưu trữ dữ liệu. Trong hệ thống tập tin, dữ liệu được lưu trữ trong các tập tin trên đĩa cứng hoặc thiết bị lưu trữ khác. Mỗi tập tin được gán một tên và có thể chứa một hoặc nhiều dữ liệu. Hệ thống tập tin cung cấp các công cụ để tạo, xóa, sửa đổi và truy cập các tập tin.

Hệ thống tập tin cũng cho phép tạo ra các thư mục để lưu trữ các tập tin và các thư mục khác giúp cho việc tổ chức và quản lý dữ liệu một cách hiệu quả hơn, nó cũng cung cấp các phân quyền truy cập để xác định người dùng nào có quyền truy cập vào tập tin hoặc thư mục nào giúp bảo vệ dữ liệu khỏi sự truy cập trái phép.

Hiện nay các cá nhân sử dụng máy tính đa số sử dụng cách lưu trữ dữ liệu dưới dạng hệ thống tập tin để lưu trữ thông tin cho riêng mình, nên nói về sự phổ biến thì cách này phổ biến nhất. Tuy nhiên, cách lưu trữ này có một số hạn chế. Nó có thể không đáp ứng tốt cho việc lưu trữ lượng dữ liệu lớn hoặc việc truy cập dữ liệu theo tốc độ cao. Nó cũng không có khả năng tìm kiếm nội dung trong tập tin hoặc thư mục một cách dễ dàng. Dữ liệu dễ bị trùng lặp và không đảm bảo tính nhất quán, tính chia sẻ dữ liệu không cao. Vì vậy, khi cần lưu trữ lượng dữ liệu lớn hoặc cần truy cập dữ liệu theo tốc độ cao, người dùng thường sử dụng các hệ thống lưu trữ khác như CSDL hay hệ thống đám mây.

## 1.7.2 Lưu trữ dữ liệu dưới dạng CSDL

Cách tiếp cận theo hướng CSDL là một phương pháp quan trọng để xử lý và truy vấn dữ liệu trong các hệ quản trị CSDL. Nó bao gồm việc sử dụng các câu lệnh truy vấn và các công cụ để truy cập và xử lý dữ liệu. Bạn cần có một *hệ quản trị CSDL* và một *ngôn ngữ truy vấn dữ liệu*. Các hệ quản trị CSDL liệu phổ biến bao gồm MySQL, Microsoft SQL Server và Oracle... và các ngôn ngữ truy vấn phổ biến bao gồm *SQL (Structured Query Language)* và *NoSQL*.

Để truy vấn dữ liệu trong CSDL, chúng ta sử dụng các câu lệnh SQL để chỉ ra dữ liệu muốn truy vấn và cách chúng được sắp xếp và trả về. Ví dụ, để truy vấn tất cả các hàng trong một bảng có tên là SINHVIEN trong một CSDL, chúng ta có thể dùng câu truy vấn sau:

```
SELECT * FROM SINHVIEN;
```



Chúng ta cũng có thể sử dụng các điều kiện để lọc dữ liệu trong câu truy vấn. Ví dụ, để truy vấn tất cả các hàng trong bảng SINHVIEN mà có trường GIOITINH bằng Nam, chúng ta có thể sử dụng câu truy vấn sau:

```
SELECT * FROM SINHVIEN WHERE GIOITINH = 'Nam';
```



Chúng ta cũng có thể sử dụng các hàm tính toán và các toán tử để xử lý dữ liệu trong câu truy vấn. Ví dụ, để tính tổng giá trị của trường DIEMTHI trong bảng KETQUA, chúng ta có thể sử dụng câu truy vấn sau:

```
SELECT SUM(DIEMTHI) FROM KETQUA;
```



Ngoài việc sử dụng câu lệnh truy vấn, chúng ta cũng có thể sử dụng các công cụ quản lý cơ sở dữ liệu như MySQLWorkBench, Azure Data Studio hoặc Oracle SQL Developer để truy cập và quản lý dữ liệu trong CSDL một cách thực tế hơn.

Việc lưu trữ dữ liệu dưới dạng CSDL sẽ khắc phục được những khuyết điểm của cách lưu trữ dưới dạng hệ thống tập tin, đó là:

1. Giảm trùng lặp thông tin ở mức thấp nhất, đảm bảo tính nhất quán và toàn vẹn dữ liệu.
2. Đảm bảo dữ liệu được truy xuất theo nhiều cách khác nhau, từ nhiều người khác nhau và nhiều ứng dụng khác nhau.
3. Dữ liệu được lưu trữ độc lập với chương trình, được quản lý tập trung.
4. Đảm bảo an toàn dữ liệu.
5. Cho phép chia sẻ cho nhiều ứng dụng sử dụng đồng thời trên một CSDL.
6. Hỗ trợ nhiều khung nhìn khác nhau về cơ sở dữ liệu.

Tuy nhiên, cũng có một số điểm cần lưu ý khi sử dụng cách tiếp cận theo hướng CSDL. Một trong số đó là việc cần phải đảm bảo rằng dữ liệu trong CSDL luôn được cập nhật và chính xác, vì việc truy vấn dữ liệu dựa trên dữ liệu sai có thể dẫn đến kết quả sai. Chúng ta cũng cần phải đảm bảo rằng các câu lệnh truy vấn được viết chính xác và không có lỗi, vì việc sử dụng câu lệnh sai có thể dẫn đến kết quả sai hoặc gây ra lỗi hệ thống.

Một số lưu ý khác khi sử dụng cách tiếp cận theo hướng CSDL:

1. Hệ quản trị CSDL có thể được triển khai trên máy chủ riêng hoặc là một dịch vụ được cung cấp bởi một nhà cung cấp dịch vụ đám mây, do đó cần có kết nối mạng đáng tin cậy để truy cập vào hệ quản trị CSDL.
2. Cần có một kế hoạch an toàn dữ liệu hợp lý để đảm bảo rằng dữ liệu không bị mất hoặc hỏng. Điều này có thể bao gồm việc sao lưu dữ liệu thường xuyên và sử dụng các kỹ thuật bảo mật để bảo vệ dữ liệu khỏi sự truy cập không hợp lệ.
3. Cần có một kế hoạch quản lý CSDL hợp lý để đảm bảo rằng hệ quản trị CSDL hoạt động tốt và không bị gián đoạn. Điều này có thể bao gồm việc theo dõi và giải quyết các vấn đề hiện tại, cũng như nâng cấp và bảo trì hệ quản trị CSDL khi cần thiết.
4. Hãy chú ý đến hiệu suất của câu lệnh truy vấn. Nếu chúng ta truy vấn dữ liệu trên lượng dữ liệu lớn, có thể cần thời gian để truy vấn hoàn tất. Chúng ta có thể sử dụng các kỹ thuật như chỉ truy vấn các trường cần thiết thay vì truy vấn toàn bộ bảng, hoặc sử dụng các chỉ mục (index) để tăng tốc độ, hiệu suất của câu truy vấn.

## 1.8 Vai trò của con người trong hệ CSDL

Với một CSDL lớn, rất nhiều người tham gia vào việc thiết kế, sử dụng và duy trì CSDL đó. Những người liên quan đến hệ CSDL được chia thành hai nhóm chính. Nhóm thứ nhất gồm những người mà công việc của họ liên quan hàng ngày đến CSDL, đó là những người quản trị CSDL, thiết kế CSDL, sử dụng CSDL, phân tích hệ thống và lập trình ứng dụng. Nhóm thứ hai gồm những người làm việc để duy trì môi trường hệ CSDL nhưng không quan tâm đến bản thân CSDL, đó là những người thiết kế và cài đặt hệ quản trị CSDL, phát triển công cụ, thao tác viên và bảo trì.

### 1.8.1 Người quản trị hệ CSDL (Database Administrator – DBA)

Trong một tổ chức có nhiều người cùng sử dụng các tài nguyên, cần phải có một người giám sát và quản lý. Trong môi trường hệ CSDL, các tài nguyên là CSDL, hệ quản trị CSDL và các phần mềm liên quan. Người quản trị hệ CSDL là người chịu trách nhiệm quản lý các tài nguyên đó. Người này chịu trách nhiệm về việc cho phép truy cập CSDL, tổ chức và hướng dẫn việc sử dụng CSDL, cấp các phần mềm và phần cứng theo yêu cầu.

## 1.8.2 Người thiết kế CSDL (Database Designer)

Người này chịu trách nhiệm xác định các dữ liệu sẽ được lưu giữ trong cơ sở, chọn các cấu trúc thích hợp để biểu diễn và lưu giữ các dữ liệu đó. Những nhiệm vụ này được thực hiện trước khi CSDL được cài đặt và phổ biến. Người thiết kế có trách nhiệm giao thiệp với những người sử dụng tương lai để nắm bắt được các yêu cầu của họ và đưa ra một thiết kế thỏa mãn các yêu cầu đó. Trong một vài trường hợp, người thiết kế đóng vai trò như người quản trị để gán quyền cho các nhân viên khác cho CSDL đã được thiết kế hoàn tất.

## 1.8.3 Người dùng cuối (End User)

Là những người mà công việc của họ đòi hỏi truy cập đến CSDL để truy vấn, cập nhật và sinh ra các báo cáo. Có thể chia những người sử dụng thành hai nhóm chính: *người sử dụng thụ động* (tức là những người sử dụng không có nhiều kiến thức về hệ CSDL) và *người sử dụng chủ động* (là những người có hiểu biết tốt về hệ CSDL).

## 1.8.4 Người phân tích hệ thống và lập trình ứng dụng

Người phân tích hệ thống xác định các yêu cầu của những người sử dụng (chủ yếu là những người sử dụng thụ động) để đặc tả các chương trình phù hợp với yêu cầu của họ. Trong khi đó, người viết chương trình ứng dụng thể hiện các đặc tả của những người phân tích thành chương trình, sau đó kiểm thử, sửa lỗi làm tài liệu và bảo trì các giao tác định sẵn.

## 1.8.5 Người thiết kế và cài đặt hệ quản trị hệ CSDL

Là những người thiết kế, cài đặt các mô đun, giao diện của hệ quản trị CSDL thành các phần mềm đóng gói. Một hệ quản trị CSDL là một hệ thống phần mềm phức tạp bao gồm nhiều thành phần (mô đun). Đó là các mô đun cài đặt từ điển dữ liệu, ngôn ngữ truy vấn, bộ xử lý giao diện, truy cập dữ liệu, kiểm tra cạnh tranh, phục hồi và an toàn.

## 1.8.6 Người phát triển công cụ

Là những người thiết kế và cài đặt các công cụ (tool), đó là các phần mềm đóng gói làm dễ việc thiết kế và sử dụng CSDL.

## 1.8.7 Các thao tác viên và người bảo trì

Là những người chịu trách nhiệm về việc chạy và bảo trì phần cứng và phần mềm của hệ thống.

## 1.9 Các thành phần của hệ quản trị CSDL

### 1.9.1 Các ngôn ngữ của hệ quản trị CSDL

Một khi việc thiết kế CSDL đã hoàn thành, cần phải chọn một hệ quản trị CSDL để cài đặt CSDL. Trong các hệ quản trị CSDL hiện nay thường có các ngôn ngữ: *ngôn ngữ định nghĩa dữ liệu (Data Definition Language – DDL)* và *ngôn ngữ thao tác dữ liệu (Data Manipulation Language – DML)*.

**Ngôn ngữ định nghĩa dữ liệu** là ngôn ngữ được sử dụng để định nghĩa các lược đồ. Hệ quản trị CSDL có một chương trình dịch ngôn ngữ DDL, nhiệm vụ của nó là xử lý các câu lệnh DDL để xác định mô tả của cấu trúc lược đồ và lưu trữ mô tả lược đồ vào từ điển của hệ quản trị CSDL.

**Ngôn ngữ thao tác dữ liệu** là ngôn ngữ được sử dụng để rút trích và cập nhật dữ liệu. Các thao tác chính gồm có lấy ra, chèn vào, loại bỏ và sửa đổi các dữ liệu. Có hai kiểu ngôn ngữ thao tác dữ liệu chính: *ngôn ngữ thao tác dữ liệu mức cao*, *ngôn ngữ thao tác dữ liệu mức thấp*.

Ngôn ngữ thao tác dữ liệu mức cao (hay còn gọi là ngôn ngữ phi thủ tục) có thể được sử dụng để diễn đạt các phép toán CSDL một cách ngắn gọn. Phần lớn các hệ quản trị CSDL cho phép nhập các lệnh của ngôn ngữ thao tác dữ liệu mức cao theo cách lặp (nghĩa là sau khi nhập một lệnh, hệ thống sẽ thực hiện lệnh đó rồi mới nhập lệnh tiếp theo) hoặc được nhúng vào một ngôn ngữ lập trình vạn năng. Trong trường hợp nhúng vào ngôn ngữ khác, các lệnh của ngôn ngữ thao tác dữ liệu phải được xác định bên trong chương trình sao cho một chương trình tiền dịch có thể nhận ra chúng và được hệ quản trị CSDL xử lý.

Ngôn ngữ thao tác CSDL mức thấp (hay còn gọi là ngôn ngữ thủ tục) phải được nhúng vào trong một ngôn ngữ lập trình vạn năng. Ngôn ngữ thao tác CSDL kiểu này thường rút ra các bản ghi hoặc các đối tượng riêng rẽ và xử lý chúng một cách riêng rẽ. Vì vậy, chúng cần phải sử dụng các cấu trúc ngôn ngữ lập trình như vòng lặp, điều kiện,... để rút ra từng bản ghi một từ một tập các bản ghi. Ngôn ngữ thao tác dữ liệu mức thấp được gọi là ngôn ngữ “một lần một bản ghi”. Các ngôn ngữ thao tác dữ liệu mức cao có thể dùng một lệnh để rút ra một lúc nhiều bản ghi nên chúng được gọi là ngôn ngữ “một lần một tập hợp”.

### 1.9.2 Các giao tiếp của hệ quản trị CSDL

Các hệ quản trị CSDL cung cấp rất nhiều loại giao diện người dùng thân thiện. Các loại giao diện chính gồm có:

- *Giao diện dựa trên bảng chọn*: Các giao diện này cung cấp cho người sử dụng danh sách các lựa chọn, gọi là bảng chọn (menu) và hướng dẫn người sử dụng diễn đạt một yêu cầu từ đầu đến cuối. Các bảng chọn làm cho người sử dụng không cần nhớ các lệnh và cú pháp của ngôn ngữ truy vấn. Các bảng chọn thả xuống đã trở thành kỹ thuật phổ biến trong các giao diện dựa trên cửa sổ. Chúng thường được sử dụng trong các giao diện quét, cho phép người sử dụng nhìn thấy nội dung của một cơ sở dữ liệu theo cách không có cấu trúc.

- *Giao diện dựa trên mẫu biểu:* Các giao diện này hiển thị một mẫu biểu cho người sử dụng. Những người sử dụng có thể điền vào tất cả các ô của mẫu biểu để nhập các dữ liệu mới hoặc họ chỉ điền vào một số ô còn hệ quản trị CSDL sẽ đưa ra các dữ liệu phù hợp cho các ô khác.
- *Giao diện đồ họa:* Hiển thị một lược đồ cho người sử dụng dưới dạng biểu đồ. Người dùng có thể thực hiện một truy vấn bằng cách thao tác trên biểu đồ. Trong nhiều trường hợp, GUI sử dụng cả các bảng chọn và các mẫu biểu. Đa số các GUI sử dụng các công cụ trỏ như chuột, phím để kích các phần của sơ đồ.
- *Ngôn ngữ tự nhiên:* Yêu cầu người dùng giao tiếp bằng ngôn ngữ cụ thể được quy định.
- *Giao diện cho người quản trị hệ thống:* Đa số các hệ quản trị CSDL có các lệnh ưu tiên, chỉ có những người quản trị hệ thống mới sử dụng các lệnh đó. Đó là các lệnh tạo ra các tài khoản (account), đặt các tham số cho hệ thống, cấp các tài khoản, thay đổi lược đồ hoặc tổ chức lại các cấu trúc lưu trữ của CSDL.

## 1.10 Mô hình dữ liệu

**Mô hình dữ liệu** là một tập hợp các khái niệm dùng để biểu diễn các cấu trúc của CSDL. Cấu trúc của một CSDL bao gồm các kiểu dữ liệu, các mối liên kết và các ràng buộc trên CSDL đó. Nhiều mô hình còn có thêm một tập hợp các phép toán cơ bản để đặc tả các thao tác trên CSDL.

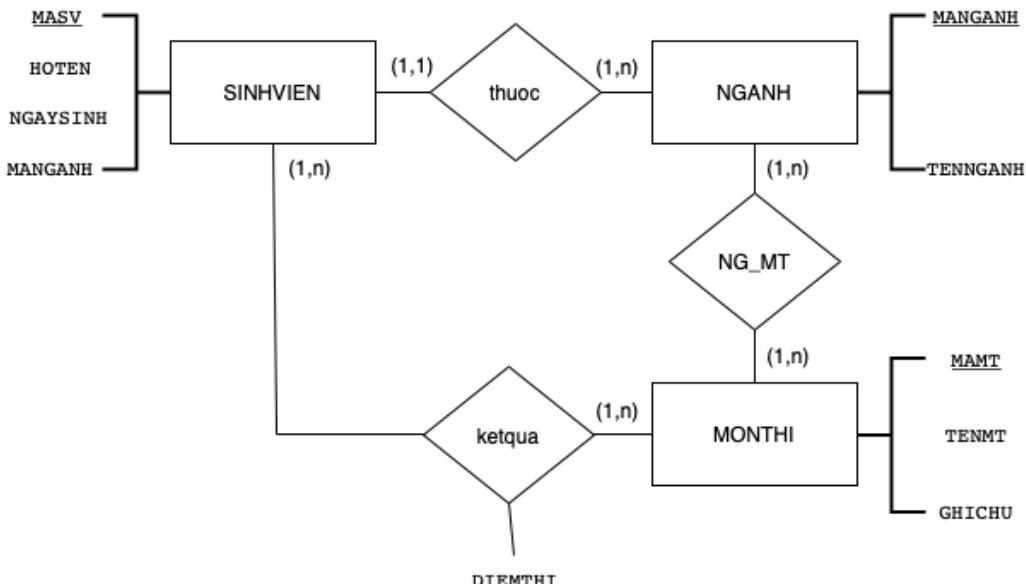
Các xử lý trên mô hình dữ liệu là các xử lý rút trích, cập nhật trên CSDL thông qua các khái niệm của mô hình dữ liệu. Các xử lý trên mô hình dữ liệu có thể là các xử lý cơ sở hoặc là các xử lý do người dùng định nghĩa.

### 1.10.1 Các loại mô hình dữ liệu

Mô hình dữ liệu được phân chia thành ba loại chính sau:

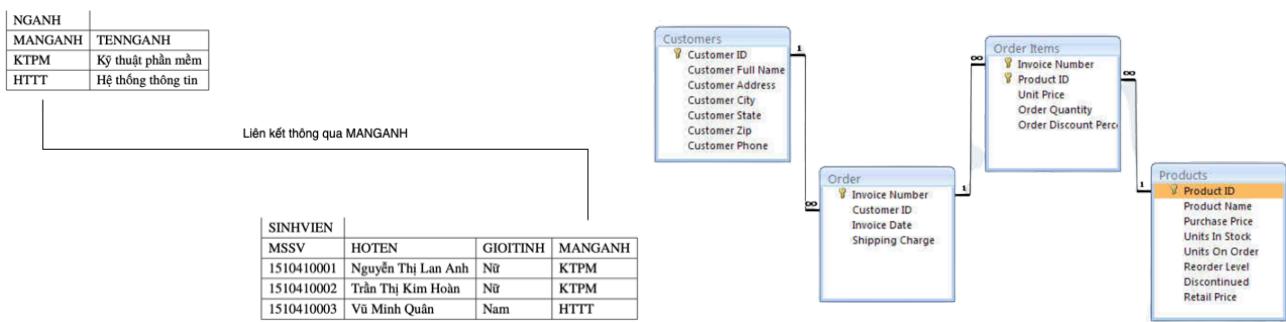
- *Mô hình dữ liệu bậc cao* (hay còn gọi là *mô hình dữ liệu mức quan niệm*) cung cấp các khái niệm gần gũi với người dùng. Các mô hình này tập trung vào bản chất logic của biểu diễn dữ liệu, nó quan tâm đến cái được biểu diễn trong cơ sở dữ liệu chứ không phải cách biểu diễn dữ liệu. Ví dụ như mô hình thực thể kết hợp, mô hình đối tượng,...
- *Mô hình dữ liệu thể hiện* (hay còn gọi là *mô hình dữ liệu mức logic*) cung cấp những khái niệm mà người sử dụng có thể hiểu được và không xa với cách tổ chức dữ liệu bên trong máy tính. Ví dụ như mô hình dữ liệu quan hệ, mô hình mạng, mô hình phân cấp,...
- *Mô hình dữ liệu bậc thấp* (hay còn gọi là *mô hình dữ liệu vật lý*) cung cấp các khái niệm mô tả chi tiết về cách dữ liệu được lưu trữ trong máy tính như thế nào.

**Mô hình thực thể kết hợp (Entity – Relationship Model)**, gọi tắt là mô hình ER là một mô hình dữ liệu bậc cao được giới thiệu bởi CHEN năm 1976. Mô hình này sử dụng các khái niệm thực thể, thuộc tính, mối liên kết, để diễn đạt các đối tượng của thế giới thực. Một *thực thể* diễn đạt một đối tượng hoặc một khái niệm của thế giới thực. Chúng ta sẽ nghiên cứu kỹ mô hình này trong Chương 2.



Hình 1.2: Ví dụ minh họa mô hình thực thể kết hợp ER

**Mô hình dữ liệu quan hệ (Relational Data Model)**, do E.F.Codd đưa ra năm 1970. Mô hình cung cấp một cấu trúc dữ liệu đơn giản và đồng bộ dựa trên khái niệm quan hệ và là cơ sở của các hệ quản trị CSDL thương mại (Oracle, DB2, MySQL, SQL Server,...). Chúng ta sẽ xem xét mô hình dữ liệu này trong nội dung của Chương 3.

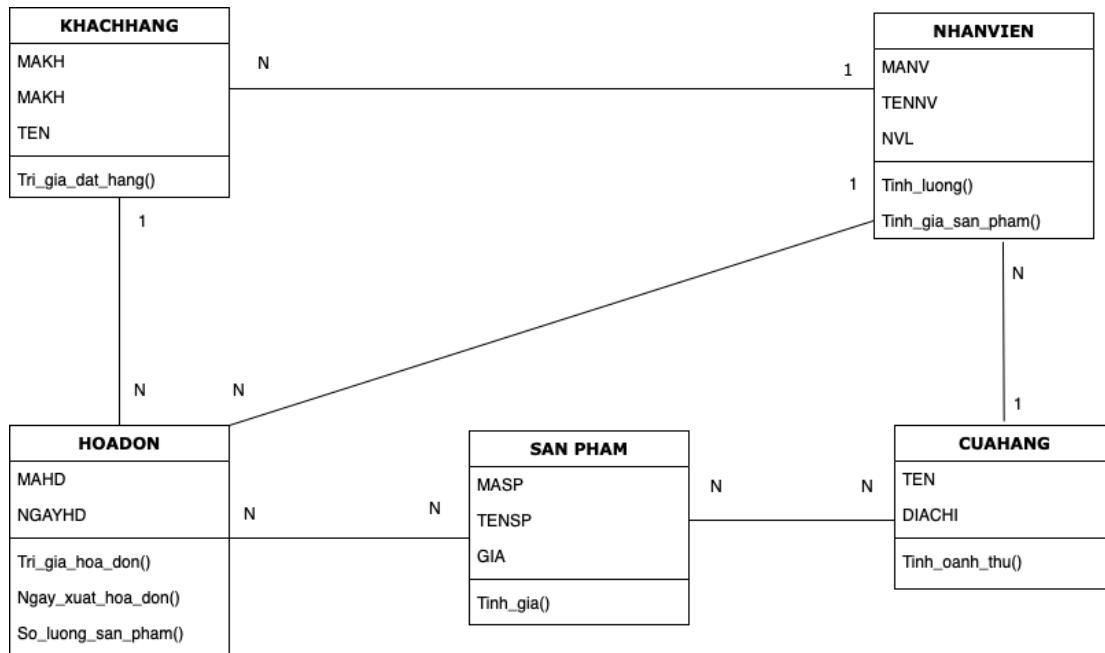


(a) Liên kết giữa các quan hệ

(b) A relationship diagram

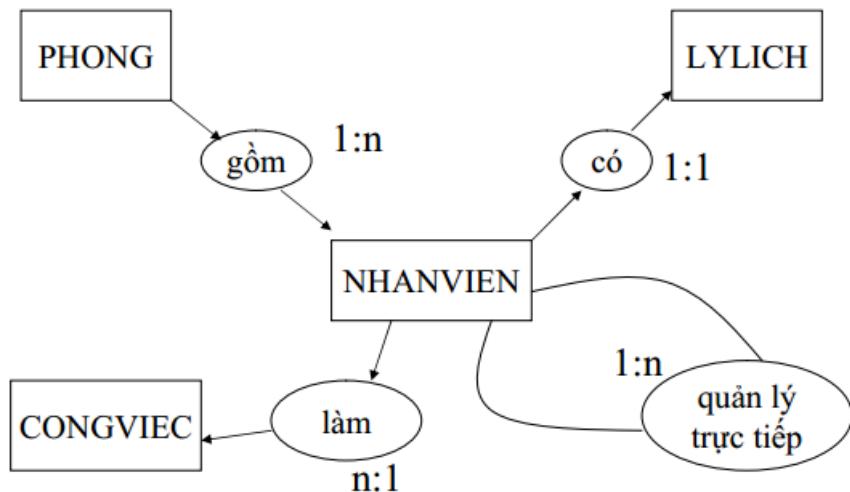
Hình 1.3: Ví dụ minh họa mô hình dữ liệu quan hệ

**Mô hình hướng đối tượng (Object-Oriented Database Model)** bắt đầu phát triển vào thập niên 90 dựa trên sự phát triển của phương pháp lập trình hướng đối tượng. Nó được thiết kế để làm việc tốt đối với những ngôn ngữ lập trình như Java, C++, C#, ... Mục đích của mô hình dữ liệu hướng đối tượng là để quản trị hiệu quả những kiểu dữ liệu phức hợp như âm thanh, hình ảnh, dữ liệu đa phương tiện,... nhằm khắc phục những hạn chế của cơ sở dữ liệu quan hệ.



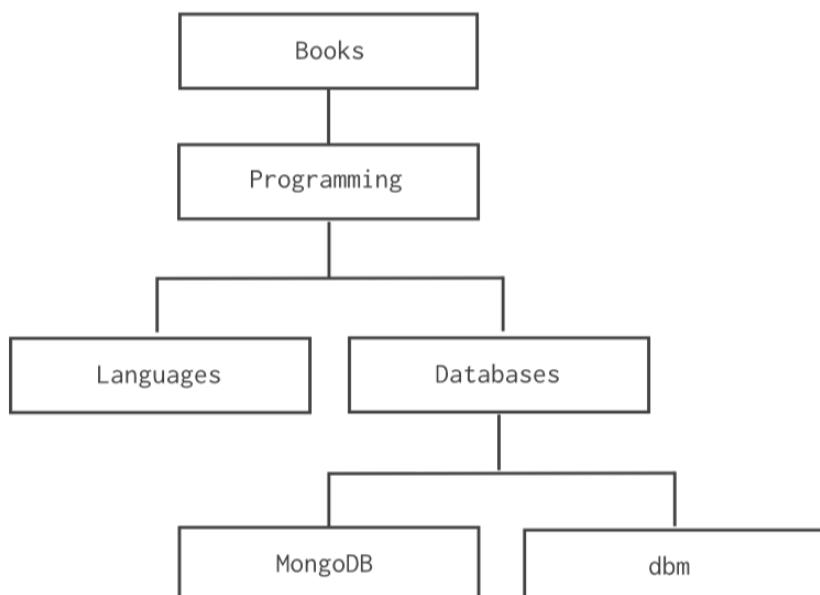
**Hình 1.4:** Ví dụ minh họa mô hình hướng đối tượng

**Mô hình dữ liệu mạng (Network Database Model)** lần đầu tiên được thực hiện bởi Honeywell vào năm 1964-1965 là mô hình được biểu diễn bởi những mẫu tin (record), loại mẫu tin, loại liên hệ giữa các mu tin và bản số của mẫu tin. Ngoài cái tên mô hình dữ liệu mang ra thì nó còn có những cái tên khác như mô hình mạng, mô hình lưới. Đây là một mô hình khá đơn giản vì chỉ dùng mối liên kết 1:1 và 1:n.



Hình 1.5: Ví dụ minh họa mô hình mạng

**Mô hình dữ liệu phân cấp (Hierarchical Database Model)** được IBM phát triển vào những năm 1960 trong đó dữ liệu được tổ chức thành cấu trúc dạng cây. Cấu trúc này tuy đơn giản nhưng không linh hoạt, vì các mối quan hệ trong cấu trúc bị ràng buộc ở dạng mối quan hệ 1:n.



Hình 1.6: Ví dụ minh họa mô hình phân cấp

### 1.10.2 Đặc trưng của một mô hình dữ liệu

- Tính ổn định khi thiết kế mô hình dữ liệu.
- Tính đơn giản có nghĩa là dễ hiểu và dễ thao tác.
- Tính dư thừa cần phải kiểm tra kỹ lưỡng.
- Tính đối xứng phải được bảo toàn.
- Có cơ sở lý thuyết vững chắc.

### 1.10.3 Phân biệt giữa các mô hình dữ liệu

Tổ chức dữ liệu theo mô hình nào là tốt nhất. Thực tế chưa có mô hình dữ liệu nào là tốt nhất. Tốt nhất phụ thuộc vào yêu cầu truy xuất và khai thác thông tin của đơn vị quản lý nó. Nó được sử dụng ở đâu và vào lúc nào là tốt nhất. Tuy nhiên, thường người ta dựa vào các tiêu chí sau để nói rằng mô hình dữ liệu tốt nhất khi:

- 1. Mục đích:* Phần lớn các mô hình dữ liệu sử dụng hệ thống ký hiệu để biểu diễn dữ liệu và làm nền tảng cho các hệ ứng dụng và ngôn ngữ thao tác dữ liệu. Các mô hình thực thể quan hệ không có hệ thống ký hiệu để xây dựng các phép toán thao tác dữ liệu, mà sử dụng để thiết kế lược đồ khái niệm, cài đặt trong một mô hình dữ liệu với một hệ quản trị cơ sở dữ liệu nào đó.
- 2. Hướng giá trị hay hướng đối tượng:* Các mô hình dữ liệu quan hệ và mô hình logic là các mô hình dữ liệu hướng giá trị. Trong các mô hình dữ liệu hướng giá trị có tính khai báo (declarativeness) và có tác động đến các ngôn ngữ được nó hỗ trợ. Các mô hình mạng, phân cấp, mô hình dữ liệu hướng đối tượng cung cấp đặc tính nhận dạng đối tượng, nên có thể xem chúng là các mô hình hướng đối tượng. Mô hình thực thể quan hệ cũng được có đặc tính nhận dạng hướng đối tượng.
- 3. Tính dư thừa:* Tất cả các mô hình dữ liệu đều có khả năng hỗ trợ lưu trữ dữ liệu vật lý và hạn chế sự dư thừa dữ liệu. Tuy nhiên các mô hình dữ liệu hướng đối tượng giải quyết sự dư thừa tốt hơn, bằng cách tạo ra sử dụng con trỏ trỏ đến nhiều vị trí khác nhau.
- 4. Giải quyết mối quan hệ nhiều – nhiều:* Phần lớn trong các mô hình cơ sở dữ liệu có chứa các mối quan hệ nhiều – nhiều, một – nhiều hay quan hệ một – một. Một quan hệ có nhiều phần tử của các quan hệ khác và ngược lại. Tuy nhiên trong mô hình dữ liệu mạng không chấp nhận mối quan hệ nhiều – nhiều.

## 1.11 Một vài ứng dụng cơ sở dữ liệu

Trong phần này ta sẽ trình bày hai ứng dụng sử dụng cách tiếp cận CSDL như sau:

1. **Ứng dụng Quản lý kết quả học tập của sinh viên.** Ứng dụng nhằm xây dựng một CSDL dùng để lưu trữ thông tin sinh viên, lớp, chuyên ngành, khoa cũng như điểm thi các môn thi của sinh viên.

Mỗi sinh viên cần lưu trữ các thông tin gồm mã số sinh viên, họ tên, địa chỉ, số điện thoại, ngày sinh, giới tính của sinh viên đó. Mỗi sinh viên có thể có nhiều số điện thoại và địa chỉ gồm có các thông tin số nhà, tên đường, quận, thành phố. Mỗi sinh viên phải thuộc về một lớp học nào đó.

Mỗi lớp lưu trữ thông tin mã lớp, số lượng lớp. Lớp phải có một sinh viên làm lớp trưởng và mỗi lớp phải thuộc một chuyên ngành nào đó.

Mỗi chuyên ngành cần lưu trữ các thông tin về mã chuyên ngành, tên chuyên ngành. Một chuyên ngành phải thuộc về sự quản lý của một khoa nào đó cụ thể.

Mỗi khoa lưu trữ thông tin về mã khoa, tên khoa, năm thành lập, phòng làm việc, số điện thoại của khoa.

Mỗi môn thi gồm có thông tin mã môn thi, tên môn thi. Mỗi chuyên ngành sẽ thi nhiều môn, và ngược lại, một môn thi cũng có thể có nhiều chuyên ngành khác nhau cùng thi. Mỗi sinh viên ứng với mỗi môn thi sẽ có một kết quả thi, nếu sinh viên vắng thi môn nào thì bị điểm 0 ở môn thi đó và cần có thông tin ghi chú là “Vắng thi” để phân biệt với một bài thi bị điểm 0.

2. **Ứng dụng Quản lý đề tài nghiên cứu khoa học.** Ứng dụng nhằm xây dựng một CSDL dùng để lưu trữ thông tin giáo viên, bộ môn, khoa cũng như các đề tài nghiên cứu khoa học mà giáo viên tham gia.

Mỗi giáo viên gồm có các thông tin về mã giáo viên, họ tên, địa chỉ, số điện thoại, ngày sinh, lương, phái. Mỗi giáo viên phải thuộc về một bộ môn cụ thể và mỗi giáo viên có thể có nhiều số điện thoại.

Mỗi bộ môn gồm có các thông tin về mã bộ môn, tên bộ môn, phòng bộ môn, điện thoại và do một giáo viên làm trưởng bộ môn, ngày nhận chức trưởng bộ môn của giáo viên đó.

Mỗi khoa cần lưu trữ thông tin về mã khoa, tên khoa, năm thành lập, phòng làm việc, số điện thoại, khoa do một giáo viên làm trưởng khoa và ngày nhận chức trưởng khoa của giáo viên đó. Một khoa có thể có nhiều bộ môn, nhưng một bộ môn chỉ thuộc về một khoa nào đó mà thôi.

Mỗi đề tài có các thông tin mã đề tài, tên đề tài, cấp quản lý, kinh phí, ngày bắt đầu, ngày kết thúc và thuộc về một chủ đề cụ thể. Mỗi chủ đề gồm có mã chủ đề, tên chủ đề. Mỗi đề tài có thể chia làm nhiều công việc.

Mỗi công việc gồm có số thứ tự, tên công việc, ngày bắt đầu, ngày kết thúc. Mỗi giảng viên có thể tham gia vào nhiều công việc cụ thể của các đề tài và mỗi công việc cũng có thể cho phép nhiều giáo viên tham gia. Khi giáo viên tham gia vào công việc thì có ghi nhận lại kết quả thực hiện công việc cũng như phụ cấp cho giáo viên.

Chúng ta sẽ sử dụng hai ví dụ này để minh họa cho hai loại mô hình thực thể kết hợp và mô hình dữ liệu quan hệ trong các chương sau của giáo trình.

## CHƯƠNG 1. Bài tập

1. Hãy định nghĩa các khái niệm: dữ liệu, cơ sở dữ liệu, hệ quản trị cơ sở dữ liệu, hệ cơ sở dữ liệu?
2. Hãy nêu một cách ngắn gọn ba hoạt động chính liên quan đến các cơ sở dữ liệu?
3. Hãy trình bày những thuận lợi trong việc quản lý dữ liệu do hướng tiếp cận cơ sở dữ liệu mang lại so với hướng tiếp cận tập tin.
4. Hãy nêu một số mô hình dữ liệu mà bạn biết?
5. Người sử dụng cơ sở dữ liệu gồm có những loại nào? Hãy cho biết vai trò của từng loại người dùng cơ sở dữ liệu?
6. Hãy tìm một vài ứng dụng thực tế có thể áp dụng cơ sở dữ liệu để quản lý dữ liệu. Với mỗi ứng dụng, hãy trình bày ngắn gọn mô tả về ứng dụng, và xác định sơ lược một vài bảng dữ liệu và thuộc tính cho các bảng dữ liệu đó (cách thức trình bày theo Hình 1.3).

# CHƯƠNG 2. MÔ HÌNH THỰC THỂ KẾT HỢP

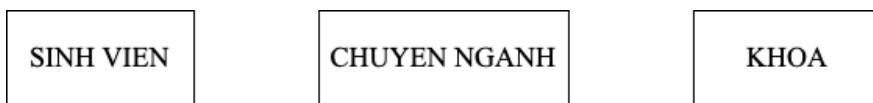
Mục đích của mô hình dữ liệu mức quan niệm là chỉ ra những quy tắc về ngữ nghĩa và mối kết hợp giữa các dữ liệu. Chương này trình bày một loại mô hình dữ liệu mức quan niệm được sử dụng rộng rãi đó là mô hình thực thể kết hợp - ER ( Entity-Relationship Model).

## 2.1 Thực thể và tập thực thể

Đối tượng được trình bày trong mô hình ER là *thực thể*. **Thực thể (entity)** là một đối tượng trong thế giới thực, có sự tồn tại độc lập. Một thực thể có thể là *thực thể cụ thể*, tức là chúng ta có thể cảm nhận được bằng các giác quan, hoặc là *thực thể trừu tượng*, tức là cái mà chúng ta không cảm nhận được bằng các giác quan nhưng có thể nhận biết được bằng nhận thức. Ví dụ: một chiếc xe ô tô, một sinh viên,... là những thực thể cụ thể. Một đơn vị công tác, một trường học... là những thực thể trừu tượng.

**Tập thực thể (entity set)** là một tập hợp các thực thể có tính chất giống nhau. Tập thực thể được ký hiệu bằng hình chữ nhật, bên trong hình chữ nhật ghi tên của tập thực thể; tên của tập thực thể có thể là danh từ hoặc là cụm danh từ. Chúng ta sẽ sử dụng khái niệm thực thể thay cho tập thực thể.

Xét trong ngữ cảnh của Ứng dụng 1 - Chương 1; sinh viên Nguyễn Văn Thành, chuyên ngành Khai phá dữ liệu, khoa Công nghệ thông tin là các thực thể cụ thể,... Trong ứng dụng này có các thực thể là SINH VIEN, CHUYEN NGANH, KHOA.

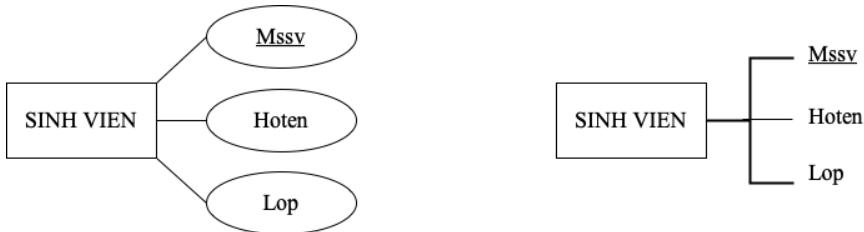


**Hình 2.1:** Ví dụ minh họa cho thực thể

## 2.2 Thuộc tính

**Thuộc tính** là các đặc trưng mô tả cho thực thể hay mối kết hợp. Ví dụ, một thực thể SINH VIEN được mô tả bằng các thuộc tính như *Họtên*, *Địa chỉ*, *SĐT*, *Ngày sinh*, *Giới tính*,... của sinh viên đó. Một thực thể cụ thể sẽ có một giá trị cụ thể cho mỗi thuộc tính đó. Ví dụ, sinh viên sv1 có các giá trị cho các thuộc tính *Họtên*, *Địa chỉ*, *SĐT*, *Ngày sinh*, *Giới tính* của nó là “*Nguyễn Văn Thành*”, “*HCM*”, “*0391234567*”, “*09/04/1990*”, “*Nam*”. Các giá trị thuộc tính mô tả mỗi thực thể sẽ trở thành một phần chính của các dữ liệu được lưu trong CSDL.

Thuộc tính được ký hiệu bằng hình elip, bên trong có ghi tên thuộc tính; tên của thuộc tính có thể là danh từ hoặc cụm danh từ. Thuộc tính cũng có thể được ký hiệu bằng cách liệt kê.



**Hình 2.2:** Ví dụ minh họa cho thuộc tính

Trong mô hình ER thuộc tính được phân thành các loại như sau:

- **Thuộc tính đơn trị** là thuộc tính không thể phân chia ra được thành các phần nhỏ hơn. Ví dụ, thuộc tính *Tuổi* của một sinh viên là một thuộc tính đơn trị.
- **Thuộc tính đa trị** là thuộc tính nhận nhiều giá trị đối với một thực thể cụ thể. Ví dụ, thuộc tính *SĐT* của một người. Một người có thể không có số điện thoại nào, người khác có thể có một số điện thoại, người khác nữa có thể có nhiều số điện thoại. Như vậy, các người khác nhau có thể có một số giá trị khác nhau cho thuộc tính *SĐT*.
- **Thuộc tính phức hợp** là thuộc tính có thể phân chia được thành các phần nhỏ hơn, biểu diễn các thuộc tính cơ bản hơn với các ý nghĩa độc lập. Ví dụ, thuộc tính *Hoten* của thực thể sinh viên có thể phân chia thành các thuộc tính *Họ* và *Tên*. Giá trị của một thuộc tính là sự kết hợp kết hợp các giá trị của các thuộc tính thành phần tạo nên nó. Việc phân chia một thuộc tính phức hợp thành các thuộc tính đơn tùy thuộc vào hoàn cảnh cụ thể.
- **Thuộc tính được lưu trữ** là các thuộc tính mà giá trị của nó được nhập vào khi cài đặt CSDL. Trong một số trường hợp, hai hay nhiều thuộc tính có giá trị liên quan đến nhau. Ví dụ, thuộc tính *Tuổi* và thuộc tính *Ngaysinh* của một người. Với một người cụ thể, ta có thể tính tuổi của họ bằng cách lấy năm hiện tại trừ đi năm của *Ngaysinh*. Thuộc tính mà giá trị của nó có thể tính được thông qua giá trị của các thuộc tính khác gọi là *thuộc tính suy diễn được*.

**Các giá trị không xác định (null values):** Trong một số trường hợp, một thực thể cụ thể có thể không có các giá trị áp dụng được cho một thuộc tính. Ví dụ, thuộc tính *SĐT* của thực thể sinh viên sẽ không có giá trị đối với các sinh viên không có số điện thoại. Trong trường hợp như vậy, ta phải tạo ra một giá trị đặc biệt gọi là giá trị không xác định (*null*). Giá trị không xác định được tạo ra khi một thuộc tính có giá trị không áp dụng được hoặc khi không biết.

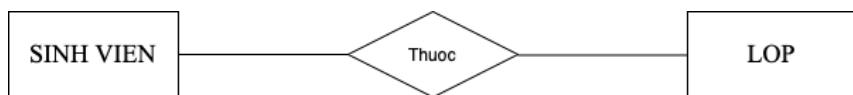
**Miền giá trị của các thuộc tính:** Mỗi thuộc tính đơn của một kiểu thực thể được kết hợp với một miền giá trị. Đó là một tập các giá trị có thể gán cho thuộc tính này đối với mỗi thực thể riêng biệt. Các miền giá trị không hiển thị trong các sơ đồ thực thể kết hợp.

## 2.3 Mối kết hợp và tập mối kết hợp

**Mối kết hợp** là sự liên kết giữa hai hay nhiều thực thể. Tập hợp các mối kết hợp tương tự nhau được gọi là **tập mối kết hợp**.

Tập mối kết hợp có thể được ký hiệu bằng hình thoi (cũng có thể sử dụng hình elip nếu như đã không sử dụng hình elip để biểu diễn thuộc tính), bên trong hình thoi có ghi tên của tập mối kết hợp; tên của tập mối kết hợp có thể là động từ hoặc cụm danh từ hoặc cụm liên từ. Tập mối kết hợp cũng có thể dùng để biểu diễn sự liên hệ giữa các tập thực thể và tập các mối kết hợp khác.

Chẳng hạn mối kết hợp giữa SINHVIEN và LOP có thể vẽ như sau:

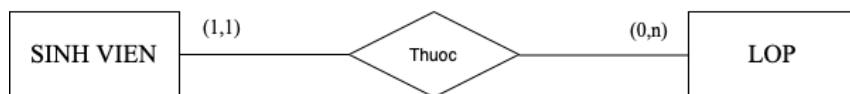


**Hình 2.3:** Ví dụ minh họa cho mối kết hợp

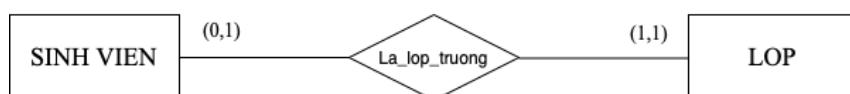
### 2.3.1 Bản số của mối kết hợp

Bản số được xác định bởi một cặp số (min – max) để quy định ràng buộc mỗi thực thể  $e$  của tập thực thể  $E$  tham gia vào tập mối kết hợp  $R$ , giá trị min thể hiện số lượng tối thiểu các thực thể thuộc tập thực thể  $E$  tham gia vào tập mối kết hợp  $R$  và giá trị max thể hiện số lượng tối đa các thực thể thuộc tập thực thể  $E$  tham gia vào tập mối kết hợp  $R$ .

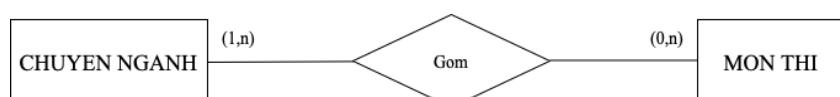
Ví dụ, xét trong ngữ cảnh của Ứng dụng 1 - Chương 1. Để biểu diễn cho mô tả “*mỗi sinh viên phải thuộc về một lớp học nào đó*”, ta có thể vẽ như sau:



Để biểu diễn cho mô tả “*lớp phải có một sinh viên là lớp trưởng*”, ta có thể vẽ như sau:



Để biểu diễn cho mô tả “*mỗi chuyên ngành gồm nhiều môn thi và một môn thi có thể có nhiều ngành thi*”, ta có thể vẽ như sau:

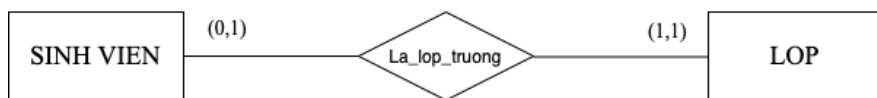


### 2.3.2 Phân loại tập mối kết hợp

Tập mối kết hợp thường gấp ba loại sau:

- *Tập mối kết hợp một - một*: Một thực thể trong tập thực thể  $A$  chỉ có thể kết hợp với tối đa một thực thể trong tập thực thể  $B$  và ngược lại một thực thể trong tập thực thể  $B$  chỉ có thể kết hợp với tối đa một thực thể trong  $A$ .

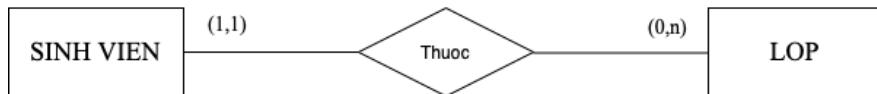
Ví dụ như một sinh viên có thể làm lớp trưởng tối đa một lớp và một lớp chỉ có duy nhất một sinh viên làm lớp trưởng.



**Hình 2.4:** Ví dụ minh họa cho tập mối kết hợp 1 - 1

- *Tập mối kết hợp một - nhiều (hoặc nhiều - một)*: Một thực thể trong tập thực thể  $A$  kết hợp với một thực thể trong tập thực thể  $B$ . Tuy nhiên, một thực thể trong tập thực thể  $B$  có thể kết hợp được với một số thực thể (không hoặc nhiều) trong  $A$ .

Ví dụ như một sinh viên chỉ thuộc về một lớp và một lớp thì có thể có nhiều sinh viên.



**Hình 2.5:** Ví dụ minh họa cho tập mối kết hợp 1 - n

- *Tập mối kết hợp nhiều - nhiều*: Một thực thể trong tập thực thể  $A$  kết hợp với một số (không hoặc nhiều) thực thể trong tập thực thể  $B$  và một thực thể trong tập thực thể  $B$  kết hợp với một số (không hoặc nhiều) thực thể trong  $A$ .

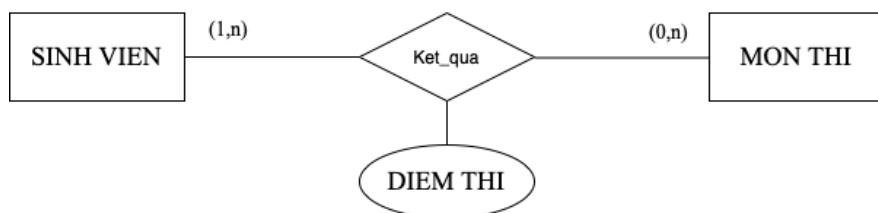
Ví dụ như một chuyên ngành có thể có nhiều môn thi và một môn thi cũng có thể được thi bởi nhiều chuyên ngành.



**Hình 2.6:** Ví dụ minh họa cho tập mối kết hợp n - n

### 2.3.3 Thuộc tính trên tập mối kết hợp

Tập mối kết hợp cũng có thể có các thuộc tính. Với tập mối kết hợp có thuộc tính thì cần đặt tên mối kết hợp sao cho có ý nghĩa. Chẳng hạn giữa tập SINHVIEN và tập MONTHI thì có thuộc tính DIEMTHI để ghi nhận kết quả điểm thi từng môn cho mỗi sinh viên.



### 2.3.4 Khóa của tập thực thể

**Khóa** là một tập các thuộc tính cho ta thông tin đầy đủ để xác định được duy nhất một thực thể trong một tập thực thể. Khóa cũng giúp xác định mối kết hợp là duy nhất trong một tập mối kết hợp. Các thuộc tính tham gia vào khóa thì gọi là *thuộc tính khóa*; các thuộc tính khóa được ký hiệu bằng dấu gạch chân liền nét phía dưới tên các thuộc tính đó.

Chẳng hạn ở ví dụ minh họa trong Hình 2.7 thì MSSV là khóa của tập thực thể SINHVIEN, MALOP là khóa của tập thực thể LOP, MACN là khóa của tập thực thể CHUYENNGANH, MAKHOA là khóa của tập thực thể KHOA và MAMT là khóa của tập thực thể MONTHI.

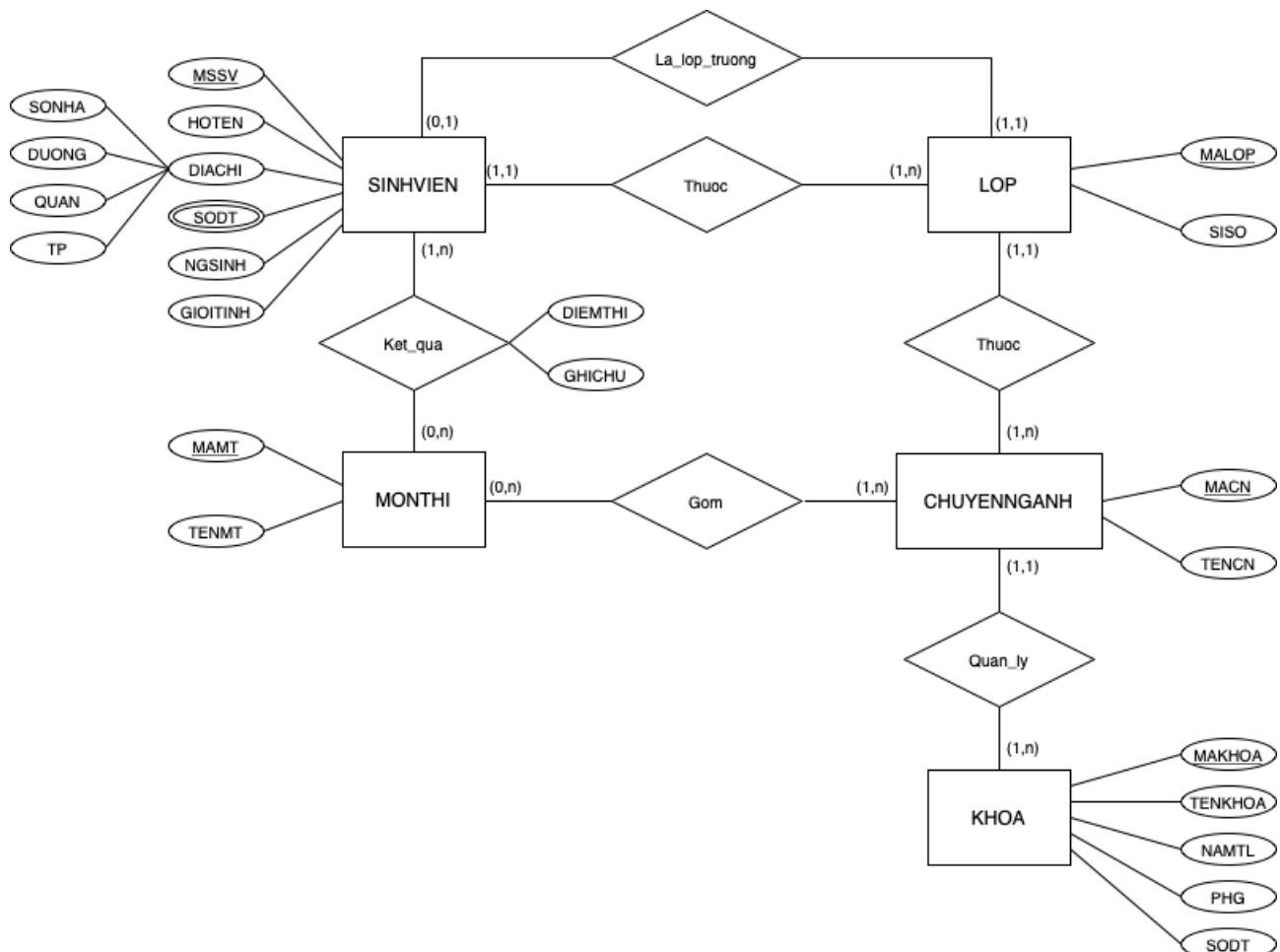
### 2.3.5 Tập thực thể yếu

Tập thực thể mà tập thuộc tính của nó không chứa khóa được gọi là **tập thực thể yếu** (tập thực thể mà tập thuộc tính của nó có chứa khóa gọi là **tập thực thể mạnh/tập thực thể chủ**).

Tập thực thể yếu thường có mối kết hợp với một tập thực thể chủ thông qua tập mối kết hợp xác định. Tập thực thể yếu luôn tham gia toàn phần vào tập mối kết hợp xác định; nhờ đó, ta có thể xác định một thực thể yếu dựa vào thực thể chủ mà nó có mối quan hệ. Tập thực thể yếu có khóa riêng phần, là tập hợp ít nhất các thuộc tính của tập thực thể yếu để xác định duy nhất một thực thể yếu trong các thực thể yếu cùng có mối kết hợp với một thực thể chủ. Tập thực thể yếu thường được ký hiệu bởi bằng hình chữ nhật được vẽ bằng nét đôi.

## 2.4 Ví dụ mô hình thực thể kết hợp

Sau đây là mô hình thực thể kết hợp cho Ứng dụng 1 - Chương 1.

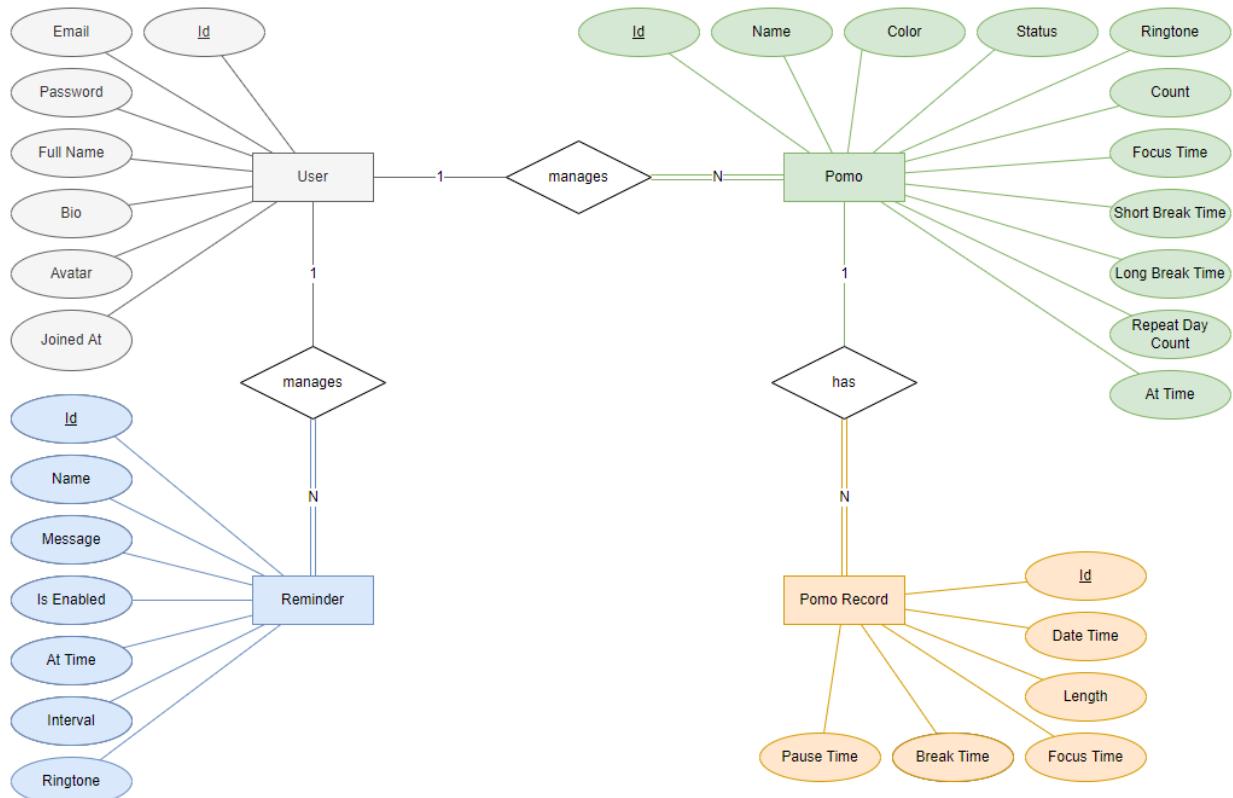


Hình 2.7: Mô hình ERD cho ứng dụng quản lý sinh viên

## 2.5 Các loại ký hiệu trong mô hình ERD

### 2.5.1 Chen's notation

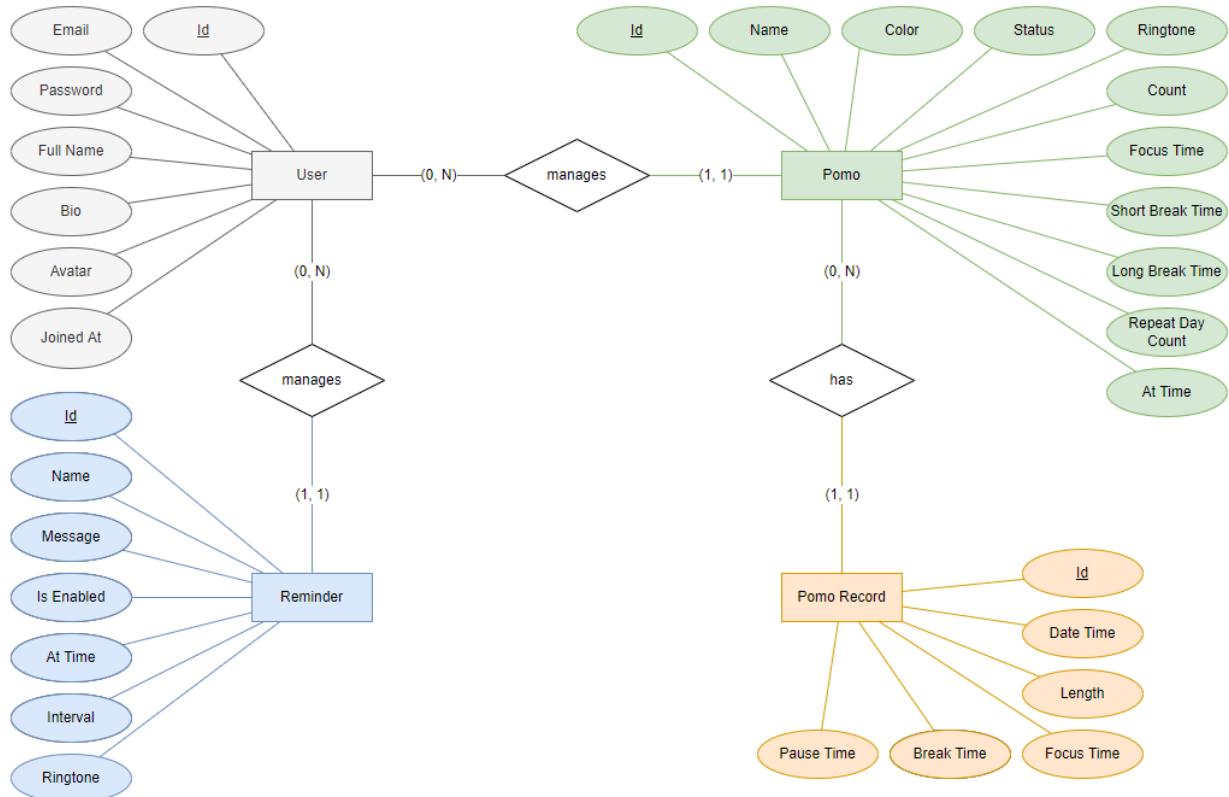
Sử dụng các ký hiệu khá đơn giản: hình chữ nhật cho thực thể, hình elip cho các thuộc tính, hình thoi cho các mối quan hệ,... Các đường nối mỗi bên sẽ có ký hiệu đơn là 1 hoặc N.



Hình 2.8: Ví dụ mô hình ERD sử dụng Chen's notations

### 2.5.2 Min-max notation

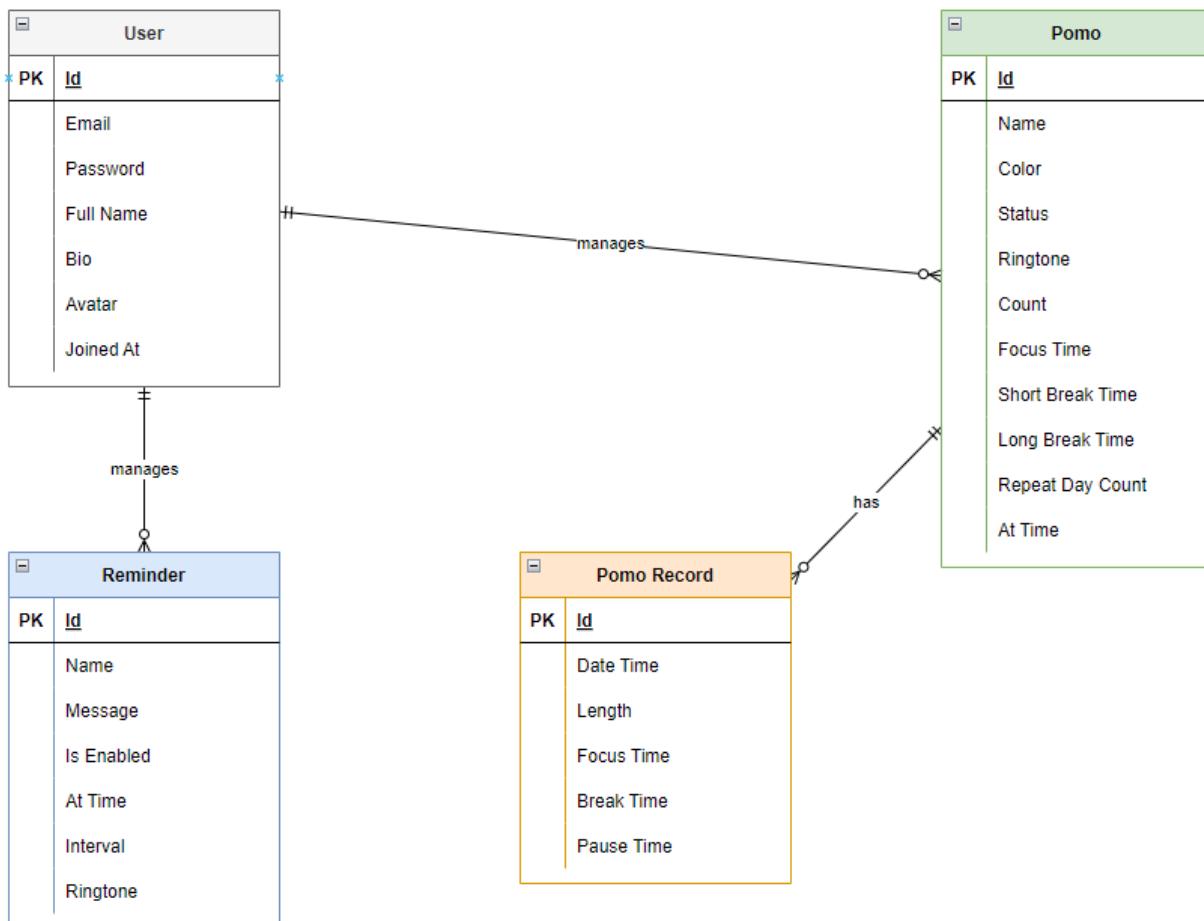
Sử dụng các ký hiệu giống với Chen's, nhưng khác ở cách đánh số. Cụ thể, mỗi bên mỗi liên kết sẽ dùng 2 số (min, max) để biểu thị số lượng thực thể.



Hình 2.9: Ví dụ mô hình Erd sử dụng Min-max notations

### 2.5.3 Crow's Foot notation

Crow's Foot notation gồm các thuộc tính và thực thể thành dạng bảng, không dùng hình thoi biểu diễn mối liên kết, và thay các con số bằng kí hiệu chân chim, nó cho phép chỉ định cả min, max ngay trong dấu chân chim. Ví dụ vòng tròn là số 0, một gạch là số 1, chân chim là N.



**Hình 2.10:** Ví dụ mô hình ERD sử dụng Crow's Foot notations

## CHƯƠNG 2. Bài tập

*Dựa vào các mô tả sơ bộ dưới đây, hãy lập mô hình thực thể kết hợp cho mỗi ứng dụng*

### 1. Ứng dụng quản lý kỳ thi tốt nghiệp trung học phổ thông

Mỗi thí sinh có một số báo danh (SOBD) duy nhất, mỗi số báo danh xác định họ và tên của thí sinh (HỌTÊN), ngày sinh (NGÀYSINH), nơi sinh (NOISINH), năm dự thi tốt nghiệp (NĂMDỰTHI). Mỗi thí sinh thuộc về một đơn vị có chức năng dạy học bậc THPT nào đó quản lý; chẳng hạn như các trường THPT, các trung tâm giáo dục thường xuyên, các trường THPT nằm trong các trường Đại học,... (các đơn vị này gọi chung là TRƯỜNG). Giả thiết thêm rằng mỗi thí sinh dự thi đều phải từ 18 tuổi trở lên.

Mỗi trường có một mã trường (MÃTRƯỜNG) duy nhất, mỗi mã trường xác định tên trường (TÊNTRƯỜNG). Mỗi môn thi có một mã môn thi (MÃMT) duy nhất, mỗi mã môn thi xác định tên môn thi (TÊNMVT).

Mỗi thí sinh ứng với mỗi môn thi sẽ có một kết quả điểm thi (ĐIỂMTHI) duy nhất, điểm thi từ 0 đến 10 và có một số lẻ đến 0.5. Kỳ thi tốt nghiệp THPT có đúng 6 môn, nếu thí sinh vắng thi môn nào thì bị điểm 0 ở môn thi đó và cần có thông tin ghi chú (GHICHU) là “Vắng thi” (nhằm phân biệt với một bài thi bị điểm 0).

Điểm xét tốt nghiệp (ĐXTN) = Tổng số điểm các bài thi/Tổng số môn thi. Một thí sinh được xem là đậu tốt nghiệp nếu không có bài thi nào bị điểm 0 và có ĐXTN từ 5.0 trở lên.

Xếp loại tốt nghiệp: Thí sinh tốt nghiệp được xếp thành ba loại là Giỏi, Khá, Trung bình theo các tiêu chuẩn sau: Loại giỏi: ĐXTN từ 8.0 điểm trở lên và không có bài thi nào dưới 7.0. Loại khá: ĐXTN từ 6.5 điểm trở lên và không có bài thi nào dưới 6.0. Loại trung bình: các trường hợp còn lại.

Ứng dụng này cần giải quyết một số vấn đề phù hợp với thực tế, chẳng hạn:

- (a). Cho biết điểm thi các môn của thí sinh có số báo danh là “15001”.
- (b). Cho biết các thí sinh có điểm tất cả các môn thi đều  $\geq 8$ .
- (c). Cho biết các thí sinh dự thi không đủ 6 môn.
- (d). Cho biết các môn thi có điểm trung bình của tất cả các thí sinh dự thi  $< 5.0$ .
- (e). Cho biết các thí sinh được xếp loại tốt nghiệp “Loại Giỏi”.
- (f). Cho biết các thí sinh có tổng số điểm các môn thi là cao nhất.
- (g). Cho biết các trường có tất cả thí sinh dự thi đều đậu tốt nghiệp.
- (h). Cho biết số lượng thí sinh rớt tốt nghiệp của từng trường.

## 2. Ứng dụng quản lý kỳ tuyển sinh vào lớp 10

Mỗi thí sinh (THISINH) có một số báo danh duy nhất (SOBD), mỗi SOBD xác định họ tên (HOTEN) và lớp chuyên (LOPCHUYEN) mà thí sinh đó đăng ký thi.

Mỗi môn (MONTHI) thi có một tên gọi duy nhất (MON). MON có giá trị là VAN (Ngữ văn), TOAN (Toán học), ANH (Tiếng Anh), LY (Vật lý), HOA (Hóa học), SINH (Sinh vật), SU (Lịch sử), DIA (địa lý), PHAP (Tiếng Pháp), TRUNG (Tiếng Trung),... Các môn chuyên có hệ số 2, các môn chung có hệ số 1. Mỗi môn thi có thêm phần ghi chú (GHICHU) để giải thích thêm về môn thi đó (GHICHU có thể bằng khoảng trắng). Miền giá trị của LOPCHUYEN trong mô tả về thí sinh phải thuộc về miền giá trị của MON trong mô tả về MONTHI.

Mỗi thí sinh (SOBD) ứng với một môn thi (MON) có một điểm thi (DIEM) duy nhất. Mỗi thí sinh có ba môn thi; trong đó hai môn chung (môn bắt buộc) là Toán học và Ngữ Văn, môn thứ ba là môn chuyên. Riêng các thí sinh dự thi vào lớp chuyên văn thi môn thứ ba là môn văn nhưng ở trình độ chuyên (có giá trị MON là VANCH), tương tự như vậy đối với các thí sinh dự thi vào lớp chuyên toán (có giá trị MON là TOANCH).

Miền giá trị của MON trong quan hệ KETQUA thuộc về miền giá trị MON trong quan hệ MONTHI. Điểm thi là số thực lấy lẻ đến 0.5. Thí sinh vắng thi môn nào thì bị điểm 0 ở môn thi đó.

Thí sinh được xem là trúng tuyển nếu thỏa đồng thời 2 điều kiện sau đây:

- Điều kiện 1: Điểm của môn chuyên phải lớn hơn hoặc bằng 5 và điểm của các môn chung phải lớn hơn hoặc bằng 2.
- Điều kiện 2: Tổng điểm thi của cả ba môn thi phải lớn hoặc bằng mức điểm tuyển, trong đó mức điểm tuyển được lấy tùy theo từng môn chuyên. Ví dụ mức điểm tuyển của lớp chuyên toán là 30 (đã nhân hệ số 2 đối với môn chuyên), mức điểm tuyển của lớp chuyên văn là 24,...

Ứng dụng cần giải quyết một số vấn đề phù hợp với thực tế, chẳng hạn:

- (a). Lập danh sách thí sinh có điểm của cả ba môn thi đều lớn hơn hoặc bằng 8.
- (b). Lập danh sách thí sinh cùng với điểm môn chuyên của thí sinh đó.
- (c). Lập danh sách các môn chuyên có thí sinh dự thi môn chuyên đó đạt điểm 10.
- (d). Lập danh sách thí sinh không thỏa điều kiện 1 theo mô tả trên.
- (e). Lập danh sách thí sinh đạt điểm môn chuyên cao nhất.
- (f). Lập danh sách thí sinh có tổng điểm thi của ba môn là cao nhất.
- (g). Lập danh sách thí sinh thỏa điều kiện 1 và mức điểm tuyển theo điều kiện 2 là 20.
- (h). Đếm số lượng thí sinh của mỗi lớp chuyên có điểm môn chuyên lớn hơn hoặc bằng 5 và điểm các môn chung đều lớn hơn hoặc bằng 2.

### 3. Ứng dụng quản lý kết quả kỳ tuyển sinh cao học

Mỗi thí sinh có một số báo danh (SOBD) duy nhất, mỗi SOBD xác định họ và tên của thí sinh (HOTEN), ngày sinh (NGAYSINH), nơi sinh (NOISINH), ngành dự thi (MANGANH), năm dự thi (NAMDUTHI); giả sử một năm chỉ tổ chức một kỳ thi tuyển sinh cao học và mỗi thí sinh chỉ đăng ký dự thi một ngành duy nhất.

Mỗi ngành dự thi có một mã ngành (MANGANH) duy nhất, mỗi mã ngành xác định tên ngành (TENNGANH).

Mỗi môn thi có một mã môn thi (MAMT) duy nhất, mỗi mã môn thi xác định tên môn thi (TENMT), tính chất môn thi – nếu là môn ngoại ngữ thì TINHCHAT là 1, ngược lại là 0. Giả sử mỗi thí sinh đều phải dự thi 3 môn: Môn cơ bản, môn cơ sở ngành và môn ngoại ngữ nào đó; tùy theo ngành thi mà sẽ có danh sách các môn cơ bản, cơ sở ngành và môn ngoại ngữ khác nhau. Có những môn thi sẽ được áp dụng cho nhiều ngành khác nhau, chẳng hạn môn căn bản có mã số là TH01 là môn cơ sở ngành cho cả 3 ngành cao học là Khoa học máy tính, Hệ thống thông tin và Kỹ thuật mạng máy tính.

Mỗi thí sinh, ứng với một môn thi sẽ có một điểm thi (DIEMTHI) duy nhất. Giả sử rằng thí sinh trúng tuyển kỳ thi tuyển sinh cao học nếu thí sinh có điểm ngoại ngữ lớn hơn hoặc bằng 5, không vắng thi môn nào và tổng điểm hai môn chuyên môn còn lại lớn hơn hoặc bằng điểm tuyển của ngành trong năm đó. Nếu thí sinh vắng thi môn nào thì môn đó bị điểm 0 và cần ghi chú thông tin là thi sinh đã “Vắng thi” môn đó.

Ứng dụng cần giải quyết một số vấn đề phù hợp với thực tế, chẳng hạn:

- (a). Lập danh sách các thí sinh dự thi của từng ngành.
- (b). Lập bảng điểm của một môn thi của tất cả thí sinh, bảng điểm tổng hợp của các thí sinh theo từng ngành.
- (c). Lập danh sách các thí sinh trúng tuyển theo từng ngành.
- (d). Các thông kê như: Số lượng thí sinh trúng tuyển theo từng ngành, số lượng thí sinh có điểm môn ngoại ngữ  $\geq 5$  của từng ngành, số lượng thí sinh có điểm môn ngoại ngữ  $\geq 5$  và tổng điểm thi hai môn chuyên môn  $\geq$  ĐIỂM CHUẨN theo từng ngành (theo mỗi năm).
- (e). Danh sách các thí sinh có vắng thi ít nhất là một môn thi nào đó.
- (f). Danh sách các thí sinh đạt điểm về ngoại ngữ ở một kỳ thi.

#### 4. Ứng dụng quản lý phân công giảng dạy của một khoa

Mỗi giảng viên có một mã cán bộ (MAGV) duy nhất, mỗi mã giảng viên xác định họ tên (HOTEN); giảng viên của khoa này có thể được phân công giảng dạy cho khoa khác.

Mỗi giảng viên trong khoa phải thuộc về một bộ môn nào đó quản lý (MABM,TENBM).

Thông tin về giảng viên cần ghi rõ đó là giảng viên cơ hữu của khoa hay là giảng viên thỉnh giảng (từ các trường bạn hoặc từ các đơn vị bạn trong trường đều được xem là giảng viên thỉnh giảng). Do một giảng viên thỉnh giảng A có thể giảng các môn cho các bộ môn khác nhau của khoa, nên giả sử rằng ngoài các bộ môn của khoa sẽ có thể một bộ môn riêng để lưu trữ các giảng viên thỉnh giảng.

Mỗi học phần có một mã học phần (MAHP) duy nhất, mỗi mã học phần xác định tên học phần (TENHP), số tín chỉ (SOTC), số tiết (SOTIET); trong đó số tiết là số nguyên được cho từ 30 đến 90. Ví dụ học phần Kỹ thuật lập trình có mã học phần là “841040”; số tín chỉ là số nguyên từ 1 đến 6 (hệ thống chỉ quản lý hệ ĐH). Các học phần cần chỉ rõ là thuộc về sự quản lý của bộ môn nào. Mỗi khoa có một mã khoa (MAKHOA) duy nhất, mỗi mã khoa xác định tên khoa (TENKHOA).

Mỗi học phần ở một học kỳ được mở một hoặc nhiều nhóm, các nhóm này được phân công cho các giảng viên giảng dạy. Mỗi năm học có 3 học kỳ được đánh số là 1,2,3. Ví dụ học kỳ 2 năm học “2021-2022”; học phần Cấu trúc dữ liệu và giải thuật mở 5 nhóm được đánh số thứ tự là nhóm 1,2,3,4,5; trong đó giảng viên Trần Văn Minh phụ trách 2 nhóm 1 và 4, còn các nhóm 2,3,5 do giảng viên Nguyễn Chí Anh phụ trách. (Ứng dụng này chỉ giới hạn trong việc phân công giảng dạy cho giảng viên; ứng dụng không quản lý thù lao giảng viên cũng như không quản lý về lý lịch của giảng viên. Kế hoạch giảng dạy được phân công cho cả năm học. Hệ thống này chỉ quản lý các lớp chính quy – còn các lớp hình thức Vừa Làm Vừa Học do có cách tổ chức riêng nên không thuộc phạm vi của ứng dụng này).

Ứng dụng cần giải quyết một số vấn đề phù hợp với thực tế, chẳng hạn:

- (a). Danh sách các học phần được mở ở học kỳ 3 của một năm học nào đó.
- (b). Danh sách các giảng viên có phân công giảng dạy ở học kỳ 1 và học kỳ 2 của một năm học nào đó.
- (c). Danh sách các giảng viên chưa có phân công giảng dạy trong một năm học.
- (d). Danh sách các học phần được mở theo từng học kỳ của một năm học nào đó; cho biết số nhóm mở của học phần đó.
- (e). Kế hoạch các nhóm lớp do các giảng viên thỉnh giảng phụ trách của một năm học nào đó.

## 5. Ứng dụng quản lý các đề tài khoa học của giảng viên trong một khoa

Mỗi giảng viên có một mã giảng viên duy nhất (MAGV), mỗi mã giảng viên xác định các thông tin: họ tên (HOTEN), mã bộ môn (MABM).

Mỗi bộ môn có một mã bộ môn (MABM) duy nhất, mỗi mã bộ môn xác định tên bộ môn (TENBM).

Mỗi đề tài có một mã đề tài duy nhất (MADETAI), mỗi mã đề tài xác định tên đề tài (TENDETAI), kinh phí thực hiện (KINHPHI), năm thực hiện (NAM). Giả sử mỗi đề tài chỉ được thực hiện đúng trong 01 năm (nếu đề tài làm trong nhiều năm thì xem như có nhiều đề tài con, và mỗi đề tài con này sẽ làm không quá 1 năm – ứng dụng này không quản lý công việc này) và có một chủ nhiệm đề tài (CHUNHIEM). Mỗi giảng viên trong một năm chỉ được làm chủ nhiệm tối đa 2 đề tài. Chủ nhiệm đề tài thuộc bộ môn nào thì xem như đề tài được tính là của bộ môn đó.

Mỗi giảng viên có thể được phân công thực hiện nhiều đề tài trong một năm (giảng viên tham gia không quá 3 đề tài) và mỗi đề tài có thể phân công cho nhiều giảng viên cùng thực hiện (không quá 4 giảng viên).

Ứng dụng cần giải quyết một số vấn đề phù hợp với thực tế, chẳng hạn:

- (a). Hãy cho biết những giảng viên có thực hiện ít nhất là một đề tài trong một năm nào đó.
- (b). Hãy cho biết những giảng viên không thực hiện đề tài nào trong một năm nào đó.
- (c). Hãy cho biết các giảng viên cùng tham gia vào một đề tài nào đó.
- (d). Liệt kê danh sách các giảng viên có tham gia vào một số đề tài nào đó.
- (e). Liệt kê các đề tài do một giảng viên nào đó làm chủ nhiệm trong một số năm nào đó.
- (f). Liệt kê các đề tài mà mỗi bộ môn thực hiện, kèm theo tên chủ nhiệm đề tài.

## 6. Ứng dụng quản lý kết quả học tập của sinh viên

Mỗi sinh viên cần quản lý các thông tin như: họ và tên (HOTENSV), ngày sinh (NGAYSINH), giới tính (GIOITINH), nơi sinh (NOISINH), hộ khẩu thường trú (TINH). Mỗi sinh viên được cấp một mã số sinh viên duy nhất (MASV) để phân biệt với mọi sinh viên khác của trường, mỗi sinh viên chỉ thuộc về một lớp học tập nào đó.

Mỗi lớp học có một mã số lớp (MALOP) duy nhất để phân biệt với tất cả các lớp học khác trong trường: có một tên gọi (TENLOP) của lớp, mỗi lớp chỉ thuộc về một khoa và có một cố vấn học tập duy nhất trong suốt cả khóa học; tất nhiên một giảng viên có thể cố vấn nhiều lớp.

Mỗi khoa có một tên gọi (TENKHOA) và một mã số duy nhất (MAKHOA) để phân biệt với các khoa khác.

Mỗi học phần có một tên gọi (TENHP) cụ thể, ứng với một số tín chỉ (SOTC) và một mã số duy nhất (MAHP) để phân biệt với các học phần khác.

Mỗi giảng viên cần quản lý các thông tin: họ và tên (HOTENGV), cấp học vị (HOCVI), thuộc một chuyên ngành (CHUYENNGANH) và được gán cho một mã số duy nhất gọi là mã giảng viên (MAGV) để phân biệt với các giảng viên khác. Mỗi giảng viên có thể dạy nhiều môn ở nhiều khoa, nhưng chỉ thuộc về sự quản lý hành chính của một khoa. Thông tin giảng viên nhằm mục đích xác định thông tin cố vấn học tập cho các lớp (không quản lý việc phân công giảng viên).

Mỗi sinh viên ứng với một học phần có một kết quả học tập (giả sử đây là điểm cuối cùng của học phần đó; nghĩa là nó đã được tổ hợp từ điểm thi và điểm quá trình).

Hệ thống cần giải quyết một số vấn đề phù hợp với thực tế, chẳng hạn:

- (a). Lập bảng điểm các học phần của một sinh viên khi biết mã số của sinh viên đó.
- (b). Lập bảng điểm của một học phần nào đó của một lớp nào đó (ngữ cảnh câu này là lớp theo nghĩa lớp cố định chứ không phải theo nghĩa nhóm lớp khi đăng ký các học phần).
- (c). Lập danh sách các thí sinh không đạt ở từng học phần theo từng học kỳ của một năm học nào đó.
- (d). Lập danh sách các sinh viên có điểm trung bình chung các môn thỏa mãn một điều kiện nào đó,...

## 7. Ứng dụng quản lý kỳ coi thi tuyển sinh đại học cao đẳng

Một hội đồng coi thi tuyển sinh đại học có nhiều địa điểm thi, mỗi điểm thi có một mã số điểm thi duy nhất (MADIEMTHI) để phân biệt với các điểm thi khác, các điểm thi được đánh số là 1,2,3,... Mỗi điểm thi xác định địa chỉ điểm thi (ĐIACHIĐIEMTHI).

Mỗi thí sinh đăng ký dự thi sẽ được cấp một số báo danh (SOBD) duy nhất để phân biệt với các thí sinh khác. Mỗi số báo danh xác định các thông tin: họ và tên (HOTEN), ngày sinh (NGAYSINH), mã ngành đăng ký dự thi (MANGANH) và số hiệu phòng thi (PHONGTHI). Mỗi thí sinh được bố trí thi tại một phòng thi duy nhất (PHONGTHI) và có một mã ngành đăng ký dự thi duy nhất (MANGANH).

Mỗi ngành có một mã ngành (MANGANH) duy nhất để phân biệt với các ngành khác, mỗi mã ngành xác định tên ngành (TENNGANH), chẳng hạn mã ngành 524802 ứng với tên ngành là Công Nghệ Thông Tin, mã ngành 52140210 ứng với ngành Khai thác Dữ liệu lớn,...

Mỗi phòng thi có một số hiệu phòng thi (PHONGTHI) duy nhất để phân biệt với các phòng thi khác, trong một phòng thi có thể có các thí sinh của nhiều ngành thi khác nhau. Mỗi phòng thi phải thuộc về một điểm thi duy nhất (MADIEMTHI).

Mỗi phòng thi có 2 cán bộ coi thi trong phòng; ngoài ra một cán bộ giám sát có nhiệm vụ giám sát nhiều phòng thi. Hội đồng thi giả sử có 1 trưởng điểm thi, 1 phó điểm thi, một số thư ký và một số phục vụ điểm thi.

Một cán bộ coi thi thuộc về một đơn vị nào đó (nếu là cán bộ từ một đơn vị ngoài trường thì ghi rõ tên của đơn vị đó: Ví dụ: Cán bộ coi thi A là giảng viên của trường Đại học Gia Định).

Hệ thống cần giải quyết một số vấn đề phù hợp với thực tế, chẳng hạn:

- (a). Lập danh sách các thí sinh trong một phòng thi.
- (b). Lập danh sách các thí sinh theo từng ngành thi.
- (c). Lập danh sách các thí sinh theo từng địa điểm thi.
- (d). Lập danh sách phân công coi thi cho từng buổi thi.
- (e). Lập một số thống kê liên quan đến số lượng như: Số lượng thí sinh ở mỗi điểm thi? Mỗi ngành thi? Số lượng giáo viên được phân công ở mỗi điểm thi?

## 8. Ứng dụng quản lý bảo trì máy tính tại nhà

Công ty tin học ABC hoạt động trong lĩnh vực bảo trì tại nhà các sự cố liên quan đến máy tính tại nhà (giả sử hệ thống chỉ quản lý các khách hàng lẻ: bảo trì lần nào thì tính phí bảo trì xong lần đó). Hệ thống cần quản lý các đối tượng sau:

Mỗi khách hàng của công ty có các thông tin: mã khách hàng (MAKH), họ tên khách hàng (HOTENKH), địa chỉ (DIACHI), số điện thoại (DIENTHOAI).

Mỗi nhân viên của công ty có các thông tin: mã nhân viên (MANV), họ tên (HOTEN). Ứng với mỗi lượt khách hàng mà công ty có thể phân công cho một hoặc nhiều nhân viên tham gia bảo trì.

Mỗi lượt bảo trì sẽ có một phiếu nghiệm thu công việc riêng, mỗi phiếu nghiệm thu công việc có một mã số nghiệm thu (SONT) duy nhất, mỗi phiếu nghiệm thu công việc thuộc về một khách hàng nào đó (tất nhiên mỗi khách hàng có thể có nhiều phiếu nghiệm thu công việc). Mỗi phiếu nghiệm thu công việc cũng cần quản lý thêm các thông tin về chi phí bảo trì (SOTIEN), ngày đến bảo trì (NGAY), nội dung bảo trì (NOIDUNG).

Sơ bộ về quy trình bảo trì máy tính cho khách hàng: Khi máy tính của khách hàng có sự cố, khách hàng sẽ điện thoại báo cho công ty thông tin sơ lược về sự cố; công ty sẽ phân công nhân viên có kỹ năng phù hợp với sự cố đó đến bảo trì; khi bảo trì xong thì đại diện các nhân viên bảo trì sẽ ký một phiếu nghiệm thu công việc với khách hàng và đồng thời thu phí bảo trì. Các phiếu nghiệm thu công việc này cùng với số tiền thu được sẽ được chuyển về cho nhân viên kế toán của công ty để tổng hợp, lưu trữ, báo cáo.

Hệ thống cần giải quyết một số vấn đề phù hợp với thực tế, chẳng hạn:

- (a). Số lượt bảo trì của từng nhân viên có trong một tháng nào đó của một năm nào đó.
- (b). Chi tiết về tình hình bảo trì máy tính của các khách hàng trong một tháng nào đó của một năm nào đó.
- (c). Tìm danh sách các nhân viên đã tham gia bảo trì cho một khách hàng nào đó tại một ngày nào đó (ngày/tháng/năm).
- (d). Tổng số tiền bảo trì thu được theo từng ngày trong một tháng nào đó.

## 9. Ứng dụng quản lý cửa hàng bán vải

Một cửa hàng kinh doanh vải với nhiều loại khác nhau, mỗi loại vải có một mã loại vải (MALOAIVAI) duy nhất dùng để phân biệt các loại vải khác, tên loại vải (TENLOAIVAI), xuất xứ (XUATXU) và thông tin khác (THONGTINKHAC) là các thông tin liên quan nếu có. Ví dụ 1 số tên loại vải như “Cotton”, “Gấm”, “Voan”,...

Cửa hàng này có nhiều cây vải, mỗi cây vải được phân biệt với nhau bởi mã cây vải (MACAYVAI), đơn giá (DONGIA) cho mỗi đơn vị tính (DONVITINH), tình trạng (TINHTRANG) và số lượng tồn (SOLUONGTON). Thuộc tính loại vải (LOAI) chứa mã loại vải (MALOAIVAI) tương ứng để cho biết mỗi cây vải thuộc về loại vải nào.

Các đối tác của cửa hàng được phân biệt với nhau bởi mã đối tác (MADOITAC), mỗi đối tác sẽ lưu thông tin về tên người đại diện, địa chỉ, điện thoại, email của người đại diện đó và thông tin về tổng tiền nợ của đối tác, tình trạng của đối tác.

Cửa hàng có nhiều đơn hàng trong một ngày, mỗi đơn hàng (DONHANG) gồm mã đơn hàng (MADONHANG), ngày lập, ngày giao dự kiến, thành tiền là tổng trị giá tiền của hóa đơn và tình trạng của đơn hàng (có giá trị “Chưa Giao” hoặc “Đã Giao”).

Mỗi đơn hàng có một chi tiết đơn hàng (CHITIETDONHANG) cho biết thông tin các mặt hàng cụ thể tương ứng cho từng đơn hàng. Gồm các thông tin về MADONHANG, MACAYVAI và số lượng mét vải được mua (SOLUONGMET) của một cây vải đó và đơn giá thành tiền tương ứng.

Hệ thống cần giải quyết một số vấn đề phù hợp với thực tế, chẳng hạn:

- (a). Cho biết thông tin của các cây vải thuộc loại vải có tên loại là “Cotton”.
- (b). Cho biết tên người đại diện và địa chỉ của đối tác mua trên 10 mét của cuộn vải có mã “CV001”.
- (c). Cho biết thông tin cây vải chưa được đối tác nào mua. Cho biết thông tin đối tác hoặc chưa mua vải hoặc chưa có đơn hàng nào được giao.
- (d). Cho biết tên người đại diện của đối tác có tổng giá trị các đơn hàng đã mua trên 30 triệu.
- (e). Cho biết có bao nhiêu cây vải (không phải số lượng tồn) thuộc loại vải có tên loại vải là “Gấm”.
- (f). Cho biết thông tin khách hàng nợ nhiều tiền nhất và có đơn hàng chưa được giao.
- (g). Cho biết thông tin đơn hàng mua tất cả các cây vải thuộc loại “Gấm” và đơn hàng này có trạng thái “Đã Giao”

## 10. Ứng dụng quản lý cửa hàng bán nước giải khát

Cửa hàng bán lẻ nước giải khát đủ loại (nước suối, rượu, nước ngọt, bia,...). Các loại nước giải khát này thuộc nhiều hiệu khác nhau (ví dụ: nước cam hiệu Tribeco và Rừng Hương). Mỗi loại nước trong mỗi hiệu có một giá bán lẻ khác nhau. Cửa hàng có một số khách quen mua nước đều đặn ở cửa hàng - đối với số khách này, cửa hàng ghi nhận tên, địa chỉ và số điện thoại. Mỗi lần khách đến mua nước, sau khi kiểm tra các mặt hàng và số lượng cần mua, cửa hàng lập một hóa đơn trong đó có ghi các thông tin về khách hàng và chi tiết các loại nước trong mỗi hiệu cùng số lượng (đơn vị tính là chai) và số tiền tương ứng. Ở cuối hóa đơn ghi tổng số tiền phải trả. Khách sẽ thanh toán và nhận hàng ở bộ phận giao hàng. Riêng đối với khách quen, có trong hồ sơ của khách hàng, thì cửa hàng chấp nhận cho lấy hàng trước (tại cửa hàng) và thanh toán hóa đơn trong vòng 3 ngày.

Cuối mỗi ngày, cửa hàng kiểm tra lượng hàng còn trong mỗi loại nước của mỗi hiệu. Nếu lượng tồn ở dưới mức tối thiểu thì cửa hàng sẽ đặt mua thêm ngày hôm sau. Lượng tồn tối thiểu này được xác định dựa trên kinh nghiệm kinh doanh của cửa hàng. Mỗi loại nước trong mỗi hiệu được cung cấp tại một nơi duy nhất gọi là đơn vị cung ứng. Đơn vị cung ứng này có thể là xí nghiệp sản xuất hay công ty cung ứng nước giải khát. Mỗi lần đặt hàng thì cửa hàng sẽ điền vào một phiếu đặt hàng trong đó có ghi ngày đặt, số lượng cho từng loại. Đơn vị cung ứng sẽ áp dụng cho những đơn giá khác nhau cho mỗi lần đặt hàng. Đơn vị cung ứng có thể giao hàng làm nhiều lần, tối đa là 3 lần trong vòng một tuần. Mỗi lần giao hàng sẽ có một phiếu giao hàng kiêm hóa đơn trong đó có chi tiết các loại nước giải khát, nhắc lại tổng lượng đặt, lượng đã giao, lượng giao đợt này, đơn giá, số tiền tương ứng cho loại đó và số tiền tổng cộng phải trả. Cửa hàng phải thanh toán ngay khi nhận hàng.

## 11. Ứng dụng quản lý một trường đại học

Trường được chia thành các trường con: Trường KHTN, Trường KHXH, Trường Công nghệ,.... Mỗi trường có một hiệu trưởng quản lý. Mỗi hiệu trưởng quản lý một trường.

Mỗi trường có nhiều khoa. Chẳng hạn, trường KHTN có các khoa Toán, Lý, Hóa,... Mỗi một khoa chỉ thuộc về một trường. Thông tin về Khoa gồm Mã khoa, tên khoa, địa chỉ, số điện thoại, tên trường.

Mỗi Khoa cung cấp nhiều môn học. Mỗi môn học gồm có Tên môn học, mã số, số đơn vị học trình, trình độ, tên Khoa.

Mỗi môn học có thể có nhiều học phần. Mỗi học phần được lưu giữ bằng các thông tin: Mã học phần, Tên môn học, Tên giáo viên dạy, học kỳ.

Mỗi khoa có nhiều giáo viên làm việc, nhưng mỗi giáo viên chỉ làm việc cho một khoa. Mỗi một khoa có một chủ nhiệm khoa, đó là một giáo viên.

Mỗi giáo viên có thể dạy nhiều nhất là 4 học phần và cũng có thể không dạy học phần nào.

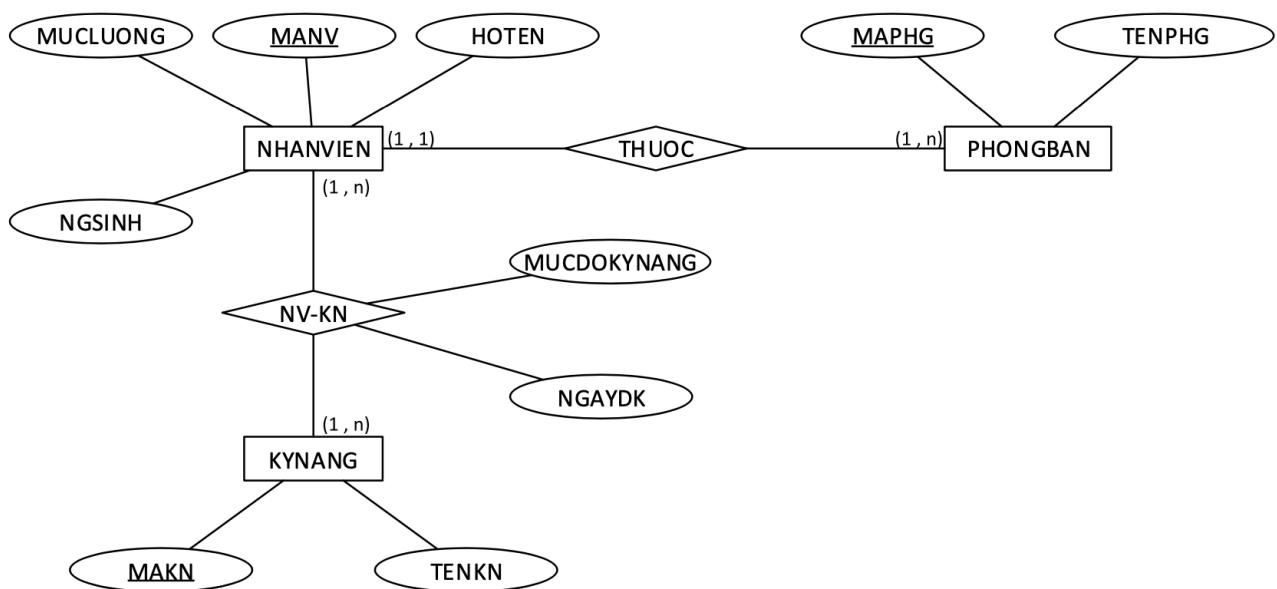
Mỗi sinh viên phải học nhiều học phần.

Mỗi một khoa có nhiều sinh viên, mỗi sinh viên chỉ thuộc về một khoa. Thông tin về mỗi sinh viên gồm: Mã sinh viên, Họ tên, địa chỉ, ngày sinh, giới tính, Lớp, Tên Khoa và chế độ đào tạo.

Mỗi sinh viên có một người giám sát (giáo viên chủ nhiệm), người đó là một giáo viên.

Sau mỗi học kỳ sẽ có một danh sách điểm để phân loại. Nó gồm các thông tin: Mã sinh viên, mã học phần, điểm bằng chữ, điểm bằng số.

12. Hãy mô tả bài toán để có được mô hình thực thể kết hợp sau đây:



# CHƯƠNG 3. MÔ HÌNH DỮ LIỆU QUAN HỆ

Mô hình dữ liệu quan hệ được Edgar F.Codd đưa ra đầu tiên vào năm 1970 , do tính chất đơn giản và được xây dựng trên nền tảng toán học vững chắc nên mô hình dữ liệu quan hệ được sử dụng rộng rãi từ thập niên 1980 cho đến tận ngày nay. Hiện tại, có nhiều hệ quản trị cơ sở dữ liệu sử dụng mô hình dữ liệu quan hệ như Access, MySQL, SQL server, Oracle,... Trong chương này sẽ trình bày những khái niệm cơ bản nhất về lý thuyết cơ sở dữ liệu quan hệ do E.F Codd đề xuất, đó là các khái niệm về quan hệ, về khóa của lược đồ quan hệ,... Những khái niệm này có vai trò quan trọng trong việc thiết kế và cài đặt các hệ cơ sở dữ liệu quan hệ và các hệ quản trị cơ sở dữ liệu.

## 3.1 Các khái niệm của mô hình dữ liệu quan hệ

**Mô hình dữ liệu quan hệ** (hay còn gọi là mô hình quan hệ) là mô hình biểu thị cơ sở dữ liệu như một tập các *quan hệ*. Mỗi quan hệ có thể được biểu diễn như một bảng giá trị, mỗi một dòng trong bảng biểu thị một tập hợp các giá trị dữ liệu liên quan với nhau. Tên bảng và tên các cột dùng để giúp giải thích ý nghĩa của các giá trị trong mỗi hàng. Mọi giá trị trong một cột đều có cùng một kiểu dữ liệu.

Cho **lược đồ quan hệ (relation schema)**  $R(A_1, A_2, \dots, A_n)$ , trong đó  $R$  là tên lược đồ quan hệ và  $A_1, A_2, \dots, A_n$  là danh sách các thuộc tính của  $R$ . Một **quan hệ (relation)**  $r$  trên lược đồ quan hệ  $R$  ký hiệu là  $r(R)$  là một tập hợp các bộ  $t_i$ ,  $r = \{t_1, t_2, \dots, t_m\}$ ; mỗi bộ  $t$  là một danh sách có thứ tự gồm  $n$  giá trị  $\langle v_1, v_2, \dots, v_n \rangle$  trong đó từng giá trị  $v_i$  thuộc tập hợp các phần tử mà thuộc tính  $A_i$  có thể nhận hoặc bằng rỗng (NULL). Mỗi quan hệ thực ra là một bảng dữ liệu hai chiều được đặt tên, có một số cột và có một số dòng dữ liệu. Mỗi dòng được gọi là một *bộ*, mỗi đầu cột được gọi là một *thuộc tính*, và bảng được gọi là một quan hệ. Kiểu dữ liệu mô tả các kiểu của dữ liệu xuất hiện trong mỗi cột gọi là một *miền*.

Ví dụ: Cho lược đồ quan hệ: SINHVIEN(MSSV, HOTEN, GIOITINH, MANGANH). Một quan hệ SINHVIEN sẽ có 4 thuộc tính là MSSV, HOTEN, GIOITINH, MANGANH và ba bộ dữ liệu. Trong đó,  $\langle 1510410001, \text{Nguyễn Thị Lan Anh}, \text{Nữ}, \text{KTPM} \rangle$  là một bộ dữ liệu của quan hệ SINHVIEN gồm các giá trị liên quan đến sinh viên “Nguyễn Thị Lan Anh”.

SINHVIEN			
MSSV	HOTEN	GIOITINH	MANGANH
1510410001	Nguyễn Thị Lan Anh	Nữ	KTPM
1510410002	Trần Thị Kim Hoàn	Nữ	KTPM
1510410003	Vũ Minh Quân	Nam	HTTT

Chúng ta dùng khái niệm “*lược đồ quan hệ*” để đề cập đến cấu trúc của một quan hệ trong khi khái niệm “*quan hệ*” đề cập đến thành phần dữ liệu của quan hệ đó.

Mỗi dòng trong quan hệ, trừ dòng tiêu đề được gọi là một **bộ (tuple)**. Mỗi bộ là một tập hợp các giá trị mô tả về một thực thể hoặc là một mối kết hợp giữa các thực thể.

Mỗi tiêu đề cột của quan hệ được gọi là **thuộc tính (attribute)**. Mỗi thuộc tính cho biết ý nghĩa của từng giá trị dữ liệu trong một bộ; đó là đặc điểm của một tập thực thể hoặc là một mối kết hợp (thuộc tính của một quan hệ không thể là thuộc tính kết hợp hay đa trị). Tên quan hệ và tên các thuộc tính giúp ta hiểu rõ hơn ngữ nghĩa của các giá trị dữ liệu trong từng dòng. Thứ tự các thuộc tính trong một quan hệ là không quan trọng. Tất cả các giá trị tại một thuộc tính có cùng kiểu dữ liệu.

Mô hình dữ liệu quan hệ thể hiện cơ sở dữ liệu bằng một tập hợp các quan hệ. Mỗi thuộc tính thuộc về một kiểu dữ liệu nào đó.

**Bậc** của một quan hệ là số thuộc tính của lược đồ quan hệ ứng với nó. Số bộ của quan hệ gọi là lực lượng của quan hệ, ký hiệu là  $|R|$ .

Một quan hệ  $r$  còn được gọi là một **thể hiện quan hệ (relation instance)** của một lược đồ quan hệ ở tại một thời điểm.

Tập hợp các giá trị mà mỗi thuộc tính  $A_i$  có thể nhận được gọi là **miền giá trị (domain)** của thuộc tính đó, ký hiệu  $Dom(A_i)$ . Một miền giá trị  $D$  là một tập hợp các giá trị nguyên tố, nghĩa là ta không đề cập đến thành phần chia nhỏ hơn nữa của giá trị đó. Người ta thường dùng kiểu dữ liệu (ví dụ ký tự, chuỗi, số nguyên, số thực, logic, ngày tháng,...) hoặc định dạng (ví dụ có 5 ký số, hoặc có 3 ký tự đầu tiên và 5 ký số sau đó,...) để thể hiện miền giá trị.

Ví dụ: Miền giá trị của MSSV, HOTEN là tập hợp các chuỗi ký tự. Thuộc tính GIOITINH trong quan hệ SINHVIEN chỉ có thể nhận các giá trị trong miền giá trị là một tập hợp có 2 phần tử {“Nam”, “Nữ”}.

**Lược đồ CSDL (database schema)**  $S$  là một tập hợp các lược đồ quan hệ  $S = \{R_1, R_2, \dots, R_m\}$ .

**Thể hiện CSDL (database instance)**  $DB$  của lược đồ CSDL  $S$  là một tập hợp các thể hiện quan hệ  $DB = \{r_1, r_2, \dots, r_m\}$  trong đó  $r_i$  là thể hiện của  $R_i$ .

## 3.2 Một số tính chất của quan hệ

### 1. Các bộ trong quan hệ là duy nhất.

Trong toán học, ta biết rằng các phần tử trong một tập hợp là không trùng nhau. Vì một quan hệ là một tập hợp các bộ nên không tồn tại hai bộ dữ liệu trùng nhau.

### 2. Thứ tự các bộ trong quan hệ là không quan trọng về mặt lý thuyết.

Cũng dựa trên lý thuyết tập hợp, các phần tử trong một tập hợp không có thứ tự, vì vậy các bộ trong một quan hệ không có một thứ tự cụ thể. Tuy nhiên, trong một tệp, các bản ghi được lưu trữ một cách vật lý trên đĩa vì vậy luôn có một thứ tự giữa các bản ghi. Thứ tự này chỉ rõ bản ghi thứ nhất, bản ghi thứ hai,... bản ghi thứ n. Một cách tương tự, khi ta biểu diễn một quan hệ như là một bảng, các hàng được hiển thị theo một thứ tự nhất định.

### 3. Thứ tự của các giá trị bên trong một bộ phụ thuộc vào thứ tự của các thuộc tính.

Theo định nghĩa quan hệ ở trên, một bộ là một danh sách có thứ tự của n giá trị. Như vậy thứ tự của các giá trị trong một bộ là quan trọng, từ đó suy ra thứ tự của các thuộc tính trong một lược đồ quan hệ cũng quan trọng. Khi đã liệt kê các thuộc tính của quan hệ theo một thứ tự nào đó thì các giá trị trong một bộ dữ liệu phụ thuộc vào thứ tự của các thuộc tính đó.

### 4. Các giá trị trong một bộ

Mỗi giá trị trong một bộ là một giá trị nguyên tử, điều đó có nghĩa là nó không phân chia được thành các thành phần trong phạm vi của mô hình quan hệ. Như vậy, trong mô hình quan hệ không cho phép có các thuộc tính phức hợp và các thuộc tính đa trị. Các thuộc tính đa trị phải được biểu diễn bằng các quan hệ còn các thuộc tính phức hợp chỉ được biểu diễn bằng các thuộc tính thành phần đơn của nó. Các giá trị của một vài thuộc tính trong một bộ cụ thể có thể không biết được hoặc không thích ứng cho nó. Trường hợp đó, người ta sử dụng một giá trị đặc biệt gọi là giá trị NULL.

### 5. Các giá trị trong một bộ

Một lược đồ quan hệ có thể được thể hiện như là một tuyên bố hoặc một khẳng định. Ví dụ lược đồ quan hệ SINHVIEN ở trên khẳng định rằng một thực thể sinh viên có một mã số, họ tên, giới tính, mã ngành. Mỗi bộ trong quan hệ được thể hiện như là một sự kiện hoặc như một thể hiện cụ thể của một khẳng định. Ngoài các quan hệ biểu diễn các sự kiện về các thực thể, một số quan hệ có thể biểu diễn các sự kiện về mối liên kết. Ví dụ, lược đồ quan hệ NG\_MT (MANGANH, MAMT) khẳng định các môn thi của từng ngành. Mỗi bộ trong quan hệ này liên kết một ngành với một môn thi của ngành đó.

Như vậy, mô hình quan hệ biểu diễn các sự kiện về thực thể và các sự kiện về liên kết dưới dạng duy nhất là các quan hệ.

### 3.3 Các khái niệm về khóa

Cho lược đồ quan hệ  $R(A_1, A_2, \dots, A_n)$  và quan hệ  $r(R)$ . Ta sẽ lần lượt định nghĩa các khái niệm liên quan đến khóa trên quan hệ  $r$  này.

- *Siêu khóa (super key)*

Ta biết rằng trong một quan hệ không tồn tại hai bộ dữ liệu giống nhau hoàn toàn trên tất cả các thuộc tính. Gọi  $S \subseteq \{A_1, A_2, \dots, A_n\}$ .  $S$  là siêu khóa của  $R$  nếu  $S$  thỏa điều kiện sau đây:

$$\forall t_1, t_2 \in r(R), t_1 \neq t_2 \text{ thì } t_1[S] \neq t_2[S]$$

Điều kiện này có nghĩa là không tồn tại hai bộ bất kỳ có giá trị giống nhau hoàn toàn trên tập  $S$ . Một quan hệ có thể có nhiều siêu khóa và có một siêu khóa mặc định là tập hợp tất cả các thuộc tính của quan hệ đó.

- *Khóa (Key)*

Gọi  $K \subseteq \{A_1, A_2, \dots, A_n\}$ .  $K$  là khóa của  $R$  nếu  $K$  thỏa 2 điều kiện sau đây:

1.  $\forall t_1, t_2 \in r(R), t_1 \neq t_2 \text{ thì } t_1[K] \neq t_2[K]$
2.  $\neg \exists K' \subset K \text{ thoả (1).}$

Khóa  $K$  của  $R$  là một siêu khóa của  $R$ , nhưng  $K$  phải thỏa mãn thêm điều kiện là nếu bỏ bất kỳ một thuộc tính  $A$  nào của  $K$  thì tập còn lại là  $K'$  sẽ không còn là siêu khóa của  $R$  nữa. Hay nói cách khác siêu khóa không chứa một siêu khóa nào khác được gọi là khóa.

Thuộc tính  $A \in K$  được gọi là *thuộc tính khóa (key attribute)*.

Ví dụ: {MSSV} là khóa của quan hệ SINHVIEN, {MSSV, HOTEN} là siêu khóa. {MSSV, HOTEN} không phải là khóa của SINHVIEN vì nếu bỏ đi thuộc tính HOTEN từ tập này thì tập còn lại là {MSSV} vẫn còn thỏa điều kiện là một siêu khóa.

- *Khóa ứng viên (candidate key)*

Một quan hệ có thể có nhiều khóa, mỗi khóa được gọi là khóa ứng viên.

Ví dụ: Quan hệ KHOA (MAKHOA, TENKHOA, NAMTL, PHG, SODT) có hai khóa ứng viên là MAKHOA và TENKHOA vì không có hai khoa nào có cùng mã hoặc có cùng tên.

- *Khóa chính (primary key)*

Trong trường hợp một lược đồ quan hệ có nhiều khóa ứng viên, thì khóa được chọn để cài đặt gọi là khóa chính (trong các phần sau, khi nói đến khóa mà không nói đến rõ hơn nữa thì ta hiểu đó là khóa chính). Khóa chính được gạch chân bằng dấu liền nét.

Ví dụ: MAKHOA được chọn là khóa chính cho quan hệ KHOA. Khi đó lược đồ quan hệ KHOA được thể hiện như sau:

KHOA(MAKHOA,TENKHOA,NAMTL,PHG,SODT)

- *Khóa ngoại (foreign key)*

Khóa ngoại cho biết mối quan hệ giữa các bộ dữ liệu trong hai quan hệ.

Cho hai lược đồ quan hệ  $R_1$  và  $R_2$  và hai quan hệ tương ứng  $r(R_1)$ ,  $r(R_2)$ . Gọi FK là một tập thuộc tính của  $R_1$  và PK là khóa chính của  $R_2$ . Ta nói FK là khóa ngoại của  $R_1$  tham chiếu đến  $R_2$  nếu hai điều kiện sau được thỏa mãn:

1. Các thuộc tính của FK tương ứng cùng miền giá trị với khóa chính PK.
2. Với mọi bộ  $t_1$  của  $r(R_1)$ , thì giá trị tại FK bằng rỗng hoặc luôn tìm thấy một bộ  $t_2$  trong quan hệ  $r(R_2)$  thỏa điều kiện  $t_1[\text{FK}] = t_2[\text{PK}]$ . Trong trường hợp FK gồm các thuộc tính khóa thì không được nhận giá trị rỗng.

Ta nói rằng bộ  $t_1$  tham chiếu đến bộ  $t_2$ .

Ví dụ: Thuộc tính MANGANH trong quan hệ:

SINHVIEN(MSSV,HOTEN,GIOITINH,MANGANH)

là khóa ngoại tham chiếu đến thuộc tính MANGANH trong quan hệ:

NGANH(MANGANH,TENNGANH)

Điều này có nghĩa là các giá trị tại thuộc tính MANGANH của bất kỳ bộ dữ liệu nào trong quan hệ SINHVIEN hoặc là bằng NULL (nếu tại thời điểm đang xét chưa biết sinh viên thuộc ngành nào) hoặc là bằng một giá trị tại một ngành nào đó trên thuộc tính là khóa chính của lược đồ quan hệ NGANH - thuộc tính MANGANH.

- *Quy ước:*

- Trong một bộ của quan hệ các thuộc tính khóa không chứa giá trị rỗng.
- Không được phép sửa đổi giá trị thuộc tính khóa của một bộ  $t$ . Nếu muốn sửa đổi giá trị thuộc tính khóa của một bộ  $t$ , người sử dụng phải huỷ bỏ bộ  $t$  và sau đó thêm một bộ  $t'$  với giá trị khóa đã được sửa đổi.

## 3.4 Chuyển mô hình thực thể kết hợp sang mô hình quan hệ

### Quy tắc 1: Chuyển tập thực thể sang quan hệ

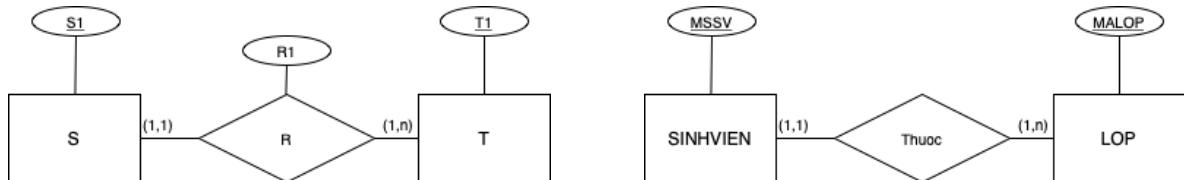
Mỗi tập thực thể mạnh  $E$  tạo thành một quan hệ  $R$  tương ứng gồm tất cả các thuộc tính đơn của  $E$ . Chọn một trong các khóa của  $E$  làm khóa chính cho  $R$ . Nếu khóa được chọn liên quan đến thuộc tính kết hợp thì các thuộc tính đơn liên quan đến thuộc tính kết hợp đó chính là khóa của  $R$ .

Ví dụ: Trong mô hình ER cho Ứng dụng quản lý sinh viên ở Chương 2, ta có các quan hệ được tạo như sau:

1. SINHVIEN(MSSV,HOTEN,SONHA,DUONG,QUAN,TP,NGSINH,GIOITINH)
2. LOP(MALOP,SISO)
3. MONTHI(MAMT,TENMT)
4. CHUYENNGANH(MACN,TENCN)
5. KHOA(MAKHOA,TENKHOA,NAMTL,PHG,SODT)

### Quy tắc 2: Tập mối kết hợp một-nhiều sang quan hệ

Gọi  $S, T$  là hai quan hệ ứng với hai tập thực thể tham gia vào tập mối kết hợp  $R$ ,  $T$  là quan hệ ứng với tập thực thể phía bên n. Bổ sung khóa chính của  $T$  vào  $S$  và tập thuộc tính này giữ vai trò khóa ngoại của  $S$ . Các thuộc tính đơn của  $R$  là thuộc tính của  $S$ .



$S(S1, \dots, T1, R1)$

$T(T1, \dots)$

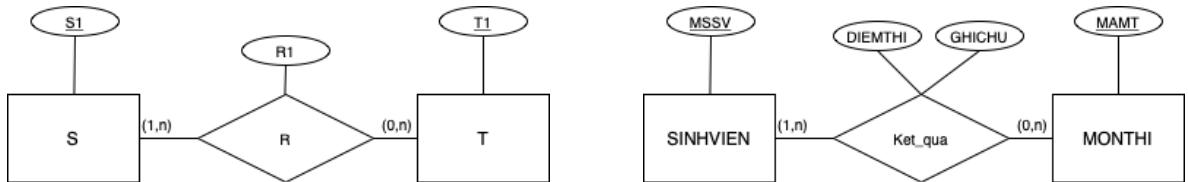
$SINHVIEN(MSSV, \dots, MALOP)$

$LOP(MALOP, \dots)$

Cách chuyển đổi tập mối kết hợp một-nhiều sang mô hình quan hệ. Trường hợp tổng quát (trái) và ví dụ minh họa (phải).

### Quy tắc 3: Tập mối kết hợp nhiều-nhiều sang quan hệ

Đối với từng tập mối kết hợp  $R$  thuộc loại này, ta tạo ra một quan hệ mới  $Q$  ứng với  $R$ . Thuộc tính của  $Q$  là tổ hợp khóa của các quan hệ ứng với các tập thực thể tham gia vào  $R$  và các thuộc tính riêng của tập mối kết hợp. Khóa của  $Q$  được xác định từ các thuộc tính khóa của quan hệ ứng với các tập thực thể tham gia vào  $R$  và dựa trên các bản số max của mỗi kết hợp.



$S(S1, \dots)$

$T(T1, \dots)$

$Q(S1, T1, R1)$

$SINHVIEN(MSSV, \dots)$

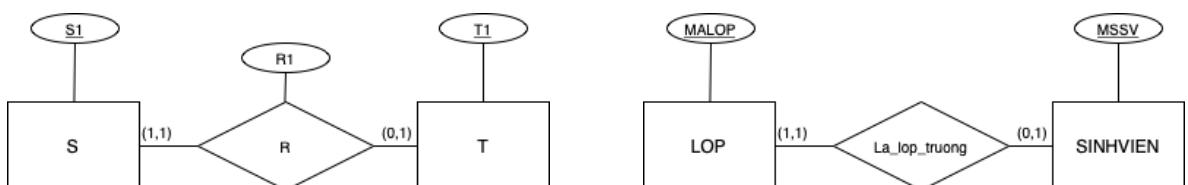
$MONTHI(MAMT, \dots)$

$KETQUA(MSSV, MAMT, DIEMTHI, GHICHU)$

Cách chuyển đổi tập mối kết hợp nhiều-nhiều sang mô hình quan hệ. Trường hợp tổng quát (trái) và ví dụ minh họa (phải).

### Quy tắc 4: Tập mối kết hợp một-một sang quan hệ

Đối với từng tập mối kết hợp  $R$  thuộc loại này, ta xác định các quan hệ  $S$  và  $T$  ứng với các tập thực thể tham gia vào  $R$ . Gọi  $S$  là quan hệ ứng với tập thực thể tham gia toàn phần vào  $R$ . Bổ sung khóa chính của  $T$  vào  $S$  và tập thuộc tính này giữ vai trò là khóa ngoại của  $S$ . Tất cả các thuộc tính đơn của  $R$  là các thuộc tính của  $S$ .



$S(S1, \dots, T1, R1)$

$T(T1, \dots)$

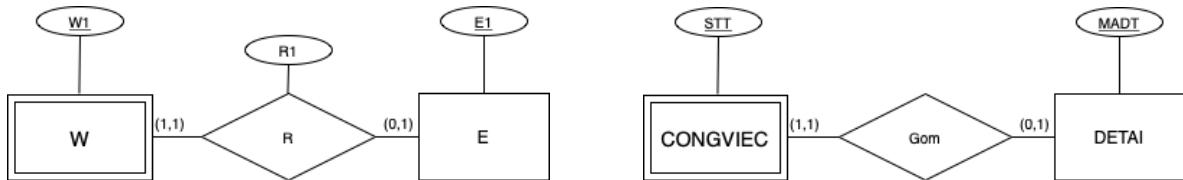
$LOP(MALOP, \dots, MSSV)$

$SINHVIEN(MSSV, \dots)$

Cách chuyển đổi tập mối kết hợp một-một sang mô hình quan hệ. Trường hợp tổng quát (trái) và ví dụ minh họa (phải).

### Quy tắc 5: Tập thực thể yếu sang quan hệ

Đối với từng tập thực thể yếu  $W$  có mối kết hợp với tập thực thể mạnh  $E$ , tạo một quan hệ  $Q$  tương ứng. Tất cả các thuộc tính đơn của  $W$  là các thuộc tính của  $Q$ . Bổ sung khóa chính của quan hệ ứng với tập thực thể mạnh  $E$  vào  $Q$  và là khóa ngoại của  $Q$ . Khóa chính của  $Q$  là sự kết hợp khóa chính của quan hệ ứng với tập thực thể mạnh  $E$  và khóa riêng phần của tập thực thể yếu  $W$ .



$Q(W1, E1, \dots, R1)$

$E(E1, \dots)$

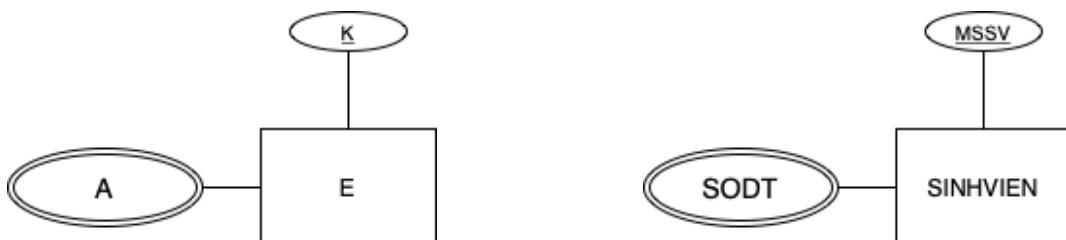
$CONGVIET(STT, MADT, \dots)$

$DETAI(MADT, \dots)$

Cách chuyển đổi tập thực thể yếu sang mô hình quan hệ. Trường hợp tổng quát (trái) và ví dụ minh họa (phải).

### Quy tắc 6: Thuộc tính đa trị sang quan hệ

Đối với từng thuộc tính đa trị  $A$ , tạo ra một quan hệ mới  $S$  gồm chính thuộc tính  $A$  và khóa chính  $K$  của quan hệ ứng với tập thực thể/tập mỗi kết hợp mà  $A$  là thuộc tính của nó. Khóa chính của  $S$  là kết hợp  $A$  và  $K$ ,  $K$  giữ vai trò là khóa ngoại trong  $S$ . Nếu thuộc tính đa trị  $A$  là thuộc tính kết hợp, ta chỉ lấy các thành phần đơn của nó.



$S(K, A)$

$E(K, \dots)$

$SV\_DT(MSSV, SODT)$

$SINHVIEN(MSSV, \dots)$

Cách chuyển đổi thuộc tính đa trị sang mô hình quan hệ. Trường hợp tổng quát (trái) và ví dụ minh họa (phải).

### 3.5 Ví dụ mô hình dữ liệu quan hệ

Sau khi hoàn thành các bước chuyển đổi mô hình ER sang mô hình quan hệ, ta có lược đồ cơ sở dữ liệu cho Ứng dụng quản lý sinh viên:

1. SINHVIEN(MSSV,HOTEN,SONHA,DUONG,QUAN,TP,NGSINH,GIOITINH,MALOP)
2. LOP(MALOP,SISO,MSSV,MACN)
3. MONTHI(MAMT,TENMT)
4. CHUYENNGANH(MACN,TENCN,MAKHOA)
5. KHOA(MAKHOA,TENKHOA,NAMTL,PHG,SODT)
6. KETQUA(MSSV,MAMT,DIEMTHI,GHICHU)
7. NG\_MT(MACN,MAMT)
8. SV\_DT(MSSV,SODT)

Lược đồ cơ sở dữ liệu cho Ứng dụng quản lý đề tài nghiên cứu khoa học:

1. GIAOVIEN(MAGV,HOTEN,LUONG,PHAI,NGSINH,DIACHI,GVQLCM,MABM)
2. GV\_DT(MAGV,DIENTHOAI)
3. BOMON(MABM,TENBM,PHG,SDT,TRUONGBM,MAKHOA,NGAYNHANCHUC)
4. KHOA(MAKHOA,TENKHOA,NAMTL,PHG,SDT,TRUONGKHOA,NGAYNHANCHUC)
5. DETAI(MADT,TENDT,KINHPHI,CAPQL,NGAYBD,NGAYKT,MACD,GVCNDT)
6. CHUDE(MACD,TENCD)
7. CONGVIEC(MADT,STT,TENCV,NGAYBD,NGAYKT)
8. THAMGIADT(MAGV,MADT,STT,PHUCAP,KETQUA)

## CHƯƠNG 3. Bài tập

1. Hãy chuyển các mô hình ER trong bài tập Chương 2 sang mô hình quan hệ. Hãy xác định khóa chính, khóa ngoại cho từng quan hệ.
2. Phòng giáo vụ tại một trường đại học muốn tin học hóa việc quản lý đăng ký học các chuyên đề của sinh viên. Sau đây là kết quả của việc phân tích thiết kế ứng dụng trên:

**SINHVIEN(MSSV, HOTEN, PHAI, NGAYSINH, DIACHI, MANGANH)**

*Tân từ:* Mỗi sinh viên có một mã số sinh viên duy nhất, một họ tên, thuộc một phái, có một ngày sinh, một địa chỉ và theo học một ngành.

**NGANH(MANGANH, TENNGANH, SOCD, TSSV)**

*Tân từ:* Mỗi ngành có một mã ngành duy nhất, có một tên ngành duy nhất. SOCD cho biết số lượng chuyên đề mà 1 sinh viên theo ngành có mã là MANGANH phải học. TSSV cho biết tổng số sinh viên đã từng theo học ngành này từ trước đến nay.

**CHUYENDE (MACD, TENCD, SSVTD)**

*Tân từ:* Mỗi chuyên đề có một mã duy nhất và có một tên duy nhất SSVTD cho biết số sinh viên tối đa có thể chấp nhận được mỗi khi có 1 lớp mở cho chuyên đề có mã là MACD.

**CD\_NGANH(MACD, MANGANH)**

*Tân từ:* Mỗi chuyên đề có thể được học bởi nhiều ngành và mỗi ngành phải học nhiều chuyên đề.

**CD\_MO(MACD, HOCKY, NAM)**

*Tân từ:* Mỗi bộ của quan hệ trên thể hiện một chuyên đề được mở ra vào một học kỳ của một năm học.

**DANGKY(MSSV, MACD, HOCKY, NAM, DIEM)**

*Tân từ:* Mỗi bộ của quan hệ trên thể hiện việc đăng ký học một chuyên đề của một sinh viên vào một học kỳ của một năm học. DIEM cho biết điểm số mà sinh viên đó đạt được khi học chuyên đề. Sinh viên chỉ được đăng ký học chuyên đề khi chuyên đề đó có mở. Sinh viên theo học mỗi ngành không được học quá 5 chuyên đề. Sinh viên không được đăng ký học quá 3 chuyên đề trong một học kỳ. Một học kỳ mở tối đa là 5 chuyên đề.

Tìm các khóa chính, khóa ngoại (nếu có) cho mỗi lược đồ quan hệ trên. Lưu ý, đối với khóa ngoại cần chỉ rõ tham chiếu đến khóa chính của lược đồ quan hệ nào.

3. Cho lược đồ CSDL sau:

NHACC(MANCC, TENNCC, ĐIACHI, ĐT)

*Tân từ:* Một nhà cung cấp có một mã nhà cung cấp, tên, địa chỉ và số điện thoại.

HANGHOA(MAHH, TENHH, ĐVT, QUYCACH, SLTON)

*Tân từ:* Cần lưu lại thông tin về tất cả các mặt hàng mà cửa hàng có mua bán: mã mặt hàng, tên hàng, đơn vị tính, quy cách, số lượng tồn.

CUNGUNG(MANCC, MAHH)

*Tân từ:* Mỗi nhà cung cấp có thể cung ứng nhiều mặt hàng khác nhau và mỗi mặt hàng cũng có thể được cung cấp bởi nhiều nhà cung cấp khác nhau, cần ghi nhận lại nhà cung cấp nào có thể cung ứng những mặt hàng gì.

DDH(SODDH, NGAYDH, MANCC)

*Tân từ:* Mỗi đơn đặt hàng có một số đơn đặt hàng duy nhất, ngày đặt hàng, đặt tại nhà cung cấp nào.

CTDDH(SODDH, MAHH, SOLUONG)

*Tân từ:* Mỗi đơn đặt hàng đặt nhiều mặt hàng khác nhau, mỗi mặt hàng ghi rõ số lượng đặt hàng. Đơn đặt hàng gửi đến một nhà cung cấp gồm các mặt hàng mà nhà cung cấp đó có thể cung ứng.

GIAOHANG(SOGH, NGAYGH, SODDH)

*Tân từ:* Mỗi phiếu giao hàng có một số phiếu, ngày giao, số đơn đặt hàng.

CTGH(SOGH, MAHH, SOLUONG)

*Tân từ:* Trong phiếu giao hàng cần ghi nhận mỗi mặt hàng có số lượng giao và đơn giá.

HOADON(SOHĐ, NGAYHĐ, TENKH)

*Tân từ:* Mỗi hóa đơn có số hóa đơn duy nhất, ngày lập hóa đơn, tên khách hàng.

CTHĐ(SOHĐ, MAHH, SOLUONG, ĐONGIA)

*Tân từ:* Cần ghi nhận đối với từng hóa đơn khách hàng đã mua những mặt hàng nào với số lượng mua, đơn giá mua là bao nhiêu.

Ngoài ra, ứng với 1 lần đặt hàng, nhà cung cấp có thể giao hàng tối đa là 3 lần và không được trễ hơn 7 ngày so với ngày đặt. Nhà cung cấp chỉ được giao các mặt hàng mà nhà cung cấp có đặt với số lượng giao không lớn hơn số lượng đặt.

Tìm các khóa chính, khóa ngoại (nếu có) cho mỗi lược đồ quan hệ trên. Lưu ý, đối với khóa ngoại cần chỉ rõ tham chiếu đến khóa chính của lược đồ quan hệ nào.

4. Xác định khóa chính, khóa ngoại cho các lược đồ CSDL sau:

**DEAN(MADA, NGAYBD, NGAYKT, TTRANG, MAPB, CHIPHI)**

*Tân từ:* Mỗi đề án có một mã số duy nhất để phân biệt với các đề án khác, có thời gian thực hiện đề án tính từ ngày bắt đầu (NGAYBD) đến ngày kết thúc (NGAYKT). Tình trạng cho biết đề án chưa thực thi, đang thực thi, thành công hay thất bại. Đề án do một phòng ban phụ trách và có một chi phí để thực hiện đề án.

**CONGDOAN(MADA, STTCD, NGAYBD, NGAYKT, MACV)**

*Tân từ:* Mỗi công đoạn của một đề án có số thứ tự (STTCD) khác nhau dùng để phân biệt với các công đoạn khác nhau trong đề án đó, có ngày bắt đầu (NGAYBD) và kết thúc (NGAYKT) công đoạn. Mỗi công đoạn triển khai duy nhất cho một công việc (MACV).

**THAMGIACD(MANV, MADA, STTCD)**

*Tân từ:* Cho biết một nhân viên tham gia vào một công đoạn trong một đề án nào.

**CONGVIEC(MACV, TENCV, CHIPHICV)**

*Tân từ:* Một công việc có một mã số duy nhất (MACV), có tên công việc (TENCV) và chi phí để triển khai công việc (CHIPHICV).

**KHANANG(MANV, MACV)**

*Tân từ:* Một nhân viên có khả năng thực hiện một số công việc nào đó mà thôi. Nhân viên chỉ được tham gia vào các công đoạn thuộc công việc mà nhân viên đó có khả năng thực hiện.

5. Xác định khóa chính, khóa ngoại cho các lược đồ CSDL sau:

**KHACH (MAKH, TENKH, DIACHIKH, DIENTHOAI)**

*Tân từ:* Mỗi khách hàng có một mã khách hàng (MAKH) duy nhất, mỗi MAKH xác định tên khách hàng (TENKH), địa chỉ (DIACHIKH), số điện thoại (DIENTHOAI).

**HANG(MAHANG,TENHANG,QUYCACH, DVTINH)**

*Tân từ:* Mỗi mặt hàng có một mã hàng (MAHANG) duy nhất, mỗi MAHANG xác định tên hàng (TENHANG), quy cách hàng (QUYCACH), đơn vị tính (DVTINH).

**DATHANG(SODH,MAHANG, SLDAT, NGAYDH, MAKH)**

*Tân từ:* Mỗi mã số đặt hàng (SODH) xác định một ngày đặt hàng (NGAYDH) và mã khách hàng tương ứng (MAKH). Biết mã số đặt hàng và mã mặt hàng thì biết được số lượng đặt hàng (SLDAT). Mỗi khách hàng trong một ngày có thể có nhiều lần đặt hàng.

**HOADON(SOHD, NGAYLAP, SODH, TRIGIAHD, NGAYXUAT)**

*Tân từ:* Mỗi hoá đơn tổng hợp có một mã số duy nhất là SOHD, mỗi hoá đơn bán hàng có thể gồm nhiều mặt hàng. Mỗi hoá đơn xác định ngày lập hoá đơn (NGAYLAP), ứng với số đặt hàng nào (SODH). Giả sử rằng hoá đơn bán hàng theo yêu cầu của chỉ một đơn đặt hàng có mã số là SODH và ngược lại, mỗi đơn đặt hàng chỉ được giải quyết chỉ trong một hoá đơn. Do điều kiện khách quan có thể công ty không giao đầy đủ các mặt hàng cũng như số lượng từng mặt hàng như yêu cầu trong đơn đặt hàng nhưng không bao giờ giao vượt ngoài yêu cầu. Mỗi hóa đơn xác định một trị giá của nhưng các mặt hàng trong hoá đơn (TRIGIAHD) và một ngày xuất kho giao hàng cho khách (NGAYXUAT).

**CHITIETHD(SOHD, MAHANG, GIABAN, SLBAN)**

*Tân từ:* Mỗi SOHD, MAHANG xác định giá bán (GIABAN) và số lượng bán (SLBAN) của một mặt hàng trong một hoá đơn.

**PHIEUTHU(SOPT, NGAYTHU, MAKH, SOTIEN)**

*Tân từ:* Mỗi phiếu thu có một số phiếu thu (SOPT) duy nhất, mỗi SOPT xác định một ngày thu (NGAYTHU) của một khách hàng có mã khách hàng là MAKH và số tiền thu là SOTIEN. Mỗi khách hàng trong một ngày có thể có nhiều số phiếu thu.

# CHƯƠNG 4. NGÔN NGỮ TRUY VẤN DỮ LIỆU

SQL (Structured Query Language) là ngôn ngữ truy vấn cơ sở dữ liệu có cấu trúc, được dùng để lưu trữ, thao tác và truy xuất dữ liệu có trong một hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) như MySQL, MS Access, Oracle, Postgres, SQL Server,...

## 4.1 Giới thiệu

SQL là viết tắt của ngôn ngữ truy vấn có cấu trúc cho phép người dùng truy cập và thao tác trên các cơ sở dữ liệu, nó đã trở thành một tiêu chuẩn của Viện Tiêu chuẩn Quốc gia Hoa Kỳ (ANSI) vào năm 1986 và của Tổ chức Tiêu chuẩn hóa Quốc tế (ISO) vào năm 1987.

SQL được sử dụng phổ biến vì nó có các ưu điểm sau:

- Cho phép người dùng tạo cơ sở dữ liệu mới
- Cho phép người dùng mô tả dữ liệu
- Cho phép người dùng thực hiện các truy vấn đối với cơ sở dữ liệu
- Cho phép nhúng trong các ngôn ngữ khác sử dụng mô-đun SQL
- Cho phép người dùng tạo và thả các cơ sở dữ liệu và bảng
- Cho phép người dùng tạo chế độ view, thủ tục lưu trữ, chức năng trong cơ sở dữ liệu
- Cho phép người dùng thiết lập quyền trên các bảng, thủ tục và view

Mặc dù SQL là một ngôn ngữ có tiêu chuẩn, nhưng nó lại có nhiều phiên bản khác nhau. Tuy nhiên, để phù hợp với tiêu chuẩn ANSI, thì các lệnh chính (chẳng hạn như SELECT, UPDATE, DELETE, INSERT, WHERE) trong tất cả các phiên bản này đều có cú pháp tương tự nhau.

## 4.2 RDBMS

**Hệ quản trị cơ sở dữ liệu quan hệ (RDBMS)** là cơ sở cho SQL, dữ liệu trong một RDBMS được lưu trữ trong các đối tượng cơ sở dữ liệu được gọi là các **bảng (table)**. Bảng này là một bộ sưu tập các mục nhập dữ liệu có liên quan và nó bao gồm nhiều **cột (column)** và **hàng (row)**.

Bảng là hình thức lưu trữ dữ liệu phổ biến và đơn giản nhất trong một cơ sở dữ liệu quan hệ. Mỗi bảng được chia thành các thực thể nhỏ gọi là các **trường (cột)**, được thiết kế để lưu trữ thông tin cụ thể về mỗi **bản ghi (dòng)** hay hàng dữ liệu trong bảng.

Sau đây là một ví dụ về một bảng SINHVIEN:

MSSV	TEN	TUOI	DIACHI	GIOITINH
12150	Doan Du	28	HCM	Nam
12151	Vuong Ngu Yen	25	Long An	Nu
12152	Nham Doanh Doanh	26	Tay Ninh	Nu

Ví dụ: Một bản ghi trong bảng SINHVIEN

MSSV	TEN	TUOI	DIACHI	GIOITINH
12151	Vuong Ngu Yen	25	Long An	Nu

Ví dụ: Một trường DIACHI trong bảng SINHVIEN

DIACHI
HCM
Long An
Tay Ninh

Một số các RDBMS phổ biến có sẵn để làm việc như:

- **MySQL** là một hệ quản trị cơ sở dữ liệu SQL mã nguồn mở, được phát triển bởi một công ty Thụy Điển - MySQL AB. MySQL hỗ trợ nhiều nền tảng khác nhau bao gồm Microsoft Windows, Linux, UNIX và Mac OS X.
- **MS SQL Server** là một hệ quản lý cơ sở dữ liệu quan hệ được phát triển bởi Microsoft Inc. Ngôn ngữ truy vấn chính của nó là *T-SQL* và *ANSI SQL*.
- **Oracle** là một hệ quản lý cơ sở dữ liệu quan hệ được phát triển bởi “Tổng công ty Oracle”. Oracle hỗ trợ tất cả các hệ điều hành bao gồm MSDOS, NetWare, UnixWare, OS/2,...
- **MS Access** là một trong những sản phẩm phổ biến nhất của Microsoft, là một phần mềm quản lý cơ sở dữ liệu cấp cơ sở. MS Access sử dụng công cụ cơ sở dữ liệu Jet, sử dụng một ngôn ngữ cụ thể (đôi khi gọi là Jet SQL).

## 4.3 Giá trị NULL

Một trường với một giá trị NULL là một trường không có giá trị. Lưu ý rằng một giá trị NULL khác với giá trị bằng không hoặc một trường có chứa khoảng trắng (space). Trường có giá trị NULL là giá trị đã để trống trong quá trình tạo bản ghi.

## 4.4 Toàn vẹn dữ liệu

Các loại sau đây của toàn vẹn dữ liệu tồn tại với mỗi RDBMS:

- *Thực thể toàn vẹn* - Không có hàng trùng lặp trong một bảng.
- *Tính toàn vẹn tham chiếu* - Các hàng không thể bị xóa, được sử dụng bởi các bản ghi khác.
- *Domain Integrity* - Thực thi kiểm tra tính hợp lệ cho một cột nhất định bằng cách hạn chế kiểu, định dạng hoặc phạm vi giá trị.
- *Tính toàn vẹn do người dùng định nghĩa* - Thực thi một số quy tắc kinh doanh cụ thể không rơi vào thực thể, miền hoặc toàn vẹn tham chiếu.

## 4.5 Ràng buộc dữ liệu

Giữa các thực thể dữ liệu tồn tại các mối quan hệ, ràng buộc lẫn nhau. Các **ràng buộc (constraint)** chính là tập các quy tắc, quy định yêu cầu dữ liệu trong cơ sở dữ liệu phải thỏa mãn. Mục đích xây dựng các ràng buộc dữ liệu là nhằm bảo đảm tính độc lập, tính toàn vẹn dữ liệu và đảm bảo tính chính xác và độ tin cậy của dữ liệu có trong CSDL. Dữ liệu lưu trữ luôn luôn hiện thực khách quan, không thừa thiếu thông tin, không mâu thuẫn thông tin. Các hệ cơ sở dữ liệu cần phải có các cơ chế cho việc mô tả các ràng buộc và quản lý các ràng buộc đã được mô tả.

Ràng buộc có thể là ở cấp độ cột hoặc cấp độ bảng. Các ràng buộc cấp độ cột chỉ được áp dụng cho một cột trong khi các ràng buộc mức bảng được áp dụng cho toàn bộ bảng. Có rất nhiều loại ràng buộc: *ràng buộc kiểu*, *ràng buộc giải tích*, *ràng buộc logic*,... đó là các khái niệm về phụ thuộc hàm, phụ thuộc đa trị, phụ thuộc kết nối.

*Ràng buộc kiểu*: Loại ràng buộc thấp nhất, mô tả tính chất của các thuộc tính khi tạo lập CSDL. Ngoài tên của thuộc tính, thuộc tính đó kiểu gì, chuỗi ký tự, kiểu số, kiểu ngày, kiểu logic... và độ dài là bao nhiêu. Ví dụ thuộc tính “SODT” là kiểu chuỗi ký tự đúng bằng 10 ký tự trong xâu. Hệ thống sẽ không chấp nhận nếu nhập vào CSDL một số điện thoại kiểu số hoặc kiểu xâu nhưng chưa đủ hoặc vượt quá 10 ký tự. Phản ứng của hệ thống hoặc là đưa ra thông báo “Dữ liệu không hợp lệ”, hoặc cắt đi những ký tự thừa.

*Ràng buộc giải tích*: Là những ràng buộc giữa các thuộc tính được biểu diễn bằng các biểu thức toán học. Ví dụ khi nhập “số lượng” và “đơn giá” của một mặt hàng, hệ thống sẽ tự động tính giá trị của thuộc tính “thành tiền” theo công thức “thành tiền” = “số lượng” x “đơn giá”. Hoặc đánh giá năng lực

học tập của một sinh viên khi nhập giá trị “điểm trung bình” của từng sinh viên vào hệ thống, hệ thống tự động đánh giá sinh viên đó có năng lực học tập là “kém”, “trung bình”, “khá” hay “giỏi”.

*Ràng buộc logic:* Mỗi quan hệ giữa các thuộc tính với nhau không phải là các ràng buộc giải tích được gọi là phụ thuộc hàm. Thuộc tính Y phụ thuộc hàm vào thuộc tính X, nghĩa là mỗi một giá trị của X xác định giá trị của Y. Ví dụ nếu giá trị của số điện thoại có thể xác định các thông tin về thuê bao có số điện thoại đó. Những ràng buộc logic có thể là ánh xạ một – một hoặc một – nhiều.

## 4.6 Các kiểu dữ liệu trong SQL

Trong mỗi bảng sẽ chứa nhiều cột và mỗi cột chúng ta phải xác định kiểu dữ liệu cho nó. Ví dụ nếu chúng ta muốn lưu tuổi thì ta sẽ lưu một số nguyên nên kiểu dữ liệu của nó là int, còn khi lưu tên thì ta sẽ lưu kiểu chuỗi varchar. Tùy vào mỗi loại mà chúng ta sẽ chọn các kiểu dữ liệu phù hợp.

Các kiểu dữ liệu trong SQL có thể được chia thành các loại sau:

- Các kiểu dữ liệu số như int, tinyint, bigint, decimal, float, real,...
- Các loại dữ liệu ngày và giờ như date, time, datetime,...
- Các kiểu dữ liệu ký tự và chuỗi như char, varchar, text,...
- Các kiểu dữ liệu chuỗi ký tự Unicode như nchar, nvarchar, ntext,...
- Các loại dữ liệu nhị phân như binary, varbinary,...
- Các loại dữ liệu khác như clob, blob, xml, cursor, table,...

Một số lưu ý khi sử dụng kiểu dữ liệu trong SQL:

- Không phải tất cả các loại dữ liệu đều được hỗ trợ bởi mọi nhà cung cấp cơ sở dữ liệu quan hệ.  
Ví dụ: Cơ sở dữ liệu Oracle không hỗ trợ dữ liệu datetime và MySQL không hỗ trợ kiểu dữ liệu clob. Vì vậy, trong khi thiết kế lược đồ cơ sở dữ liệu và viết các truy vấn sql, hãy đảm bảo kiểm tra xem các loại dữ liệu có được hỗ trợ hay không.
- Các loại dữ liệu được liệt kê ở trên không bao gồm tất cả các loại dữ liệu, đây là các loại dữ liệu được sử dụng phổ biến nhất. Một số nhà cung cấp cơ sở dữ liệu quan hệ có các loại dữ liệu riêng.  
Ví dụ: Microsoft SQL Server có các loại dữ liệu money và smallmoney nhưng vì nó không được hỗ trợ bởi các nhà cung cấp cơ sở dữ liệu phổ biến khác.
- Mỗi nhà cung cấp cơ sở dữ liệu quan hệ đều có giới hạn kích thước tối đa cho các kiểu dữ liệu khác nhau.

## Kiểu dữ liệu số (Numeric Data Types)

Danh sách các kiểu dữ liệu dạng số trong MySQL:

Kiểu dữ liệu	Mô tả
BIGINT	Một số nguyên với kích cỡ lớn, có thể là signed hoặc unsigned. Nếu có dấu, thì dãy giá trị có thể là từ -9.223.372.036.854.775.808 tới 9.223.372.036.854.775.807, nếu không dấu thì dãy giá trị là từ 0 tới 18.446.744.073.709.551.615, bạn có thể xác định độ rộng lên tới 20 chữ số.
INT	Một số nguyên với kích cỡ thông thường, có thể là signed hoặc unsigned. Nếu có dấu, thì dãy giá trị có thể là từ -2.147.483.648 tới 2.147.483.647, nếu không dấu thì dãy giá trị là từ 0 tới 4.294.967.295, bạn có thể xác định một độ rộng lên tới 11 chữ số.
MEDIUMINT	Một số nguyên với kích cỡ trung bình, có thể là signed hoặc unsigned. Nếu có dấu, thì dãy giá trị có thể là từ -8.388.608 tới 8.388.607, nếu không dấu thì dãy giá trị là từ 0 tới 16.777.215, bạn có thể xác định một độ rộng lên tới 9 chữ số.
SMALLINT	Một số nguyên nhỏ, có thể là signed hoặc unsigned. Nếu có dấu, thì dãy giá trị có thể là từ -32.768 đến 32.767, nếu không dấu thì dãy giá trị là từ 0 đến 65.535, bạn có thể xác định một độ rộng lên tới 5 chữ số.
TINYINT	Một số nguyên rất nhỏ, có thể là signed hoặc unsigned. Nếu có dấu, thì dãy giá trị có thể là từ -128 đến 127, nếu không dấu thì dãy giá trị là từ 0 đến 255, bạn có thể xác định một độ rộng lên tới 4 chữ số.
BIT	Kiểu giá trị bit. Số lượng bit trên mỗi giá trị được chỉ định rõ về kích thước. Tham số kích thước có thể giữ giá trị từ 1 đến 64. Giá trị mặc định là 1.
BOOL	Số 0 được coi là sai, các giá trị khác 0 được coi là đúng. BOOLEAN là đồng nghĩa với BOOL.
DOUBLE(N,D)	Một số thực dấu chấm động không dấu. Bạn có thể định nghĩa độ dài hiển thị (N) và số vị trí sau dấu phẩy (D). Mặc định là (16,4) với 4 là số vị trí sau dấu phẩy và 16 là số chữ số (bao gồm các phần thập phân). Phần thập phân có thể lên tới 53 vị trí sau dấu phẩy. REAL là đồng nghĩa với DOUBLE.
FLOAT(N,D)	Một số thực dấu chấm động không dấu. Bạn có thể định nghĩa độ dài hiển thị (N) và số vị trí sau dấu phẩy (D). Mặc định là (10,2) với 2 là số vị trí sau dấu phẩy và 10 là số chữ số (bao gồm các phần thập phân). Phần thập phân có thể lên tới 24 vị trí sau dấu phẩy đối với một số FLOAT.
DECIMAL(N,D)	Một kiểu khác của dấu chấm động không dấu. Mỗi chữ số thập phân chiếm 1 byte. Việc định nghĩa độ dài hiển thị (N) và số vị trí sau dấu phẩy (D) là bắt buộc. NUMERIC là một từ đồng nghĩa cho DECIMAL. NUMERIC là đồng nghĩa với DECIMAL.

## Kiểu dữ liệu ngày và giờ (Date and Time Data Types)

Kiểu dữ liệu Date và Time được phân loại thành:

Kiểu dữ liệu	Mô tả
DATETIME	Kết hợp ngày và giờ theo định dạng YYYY-MM-DD HH:MM:SS, giữa 1000-01-01 00:00:00 và 9999-12-31 23:59:59. Ví dụ, 3:30 chiều ngày 25 tháng 02, năm 2023 sẽ được lưu ở dạng 2023-02-25 15:30:00.
DATE	Một ngày trong định dạng YYYY-MM-DD, giữa 1000-01-01 và 9999-12-31. Ví dụ, ngày 25 tháng 02 năm 2023 sẽ được lưu ở dạng 2023-02-25
TIME	Lưu trữ thời gian theo định dạng HH: MM: SS.
TIMESTAMP	Mốc thời gian lúc nửa đêm. Kiểu này giống như định dạng DATETIME trước đó chỉ không có dấu gạch nối giữa các số; ví dụ: 3:30 chiều ngày 30 tháng 02 năm 2023 sẽ được lưu giữ là 20230230153000 (YYYYMMDDHHMMSS).
YEAR(N)	Lưu 1 năm trong định dạng 2 chữ số hoặc 4 chữ số. Nếu độ dài được xác định là 2 (ví dụ: YEAR(2)), YEAR có thể từ 1970 tới 2069 (70 tới 69). Nếu độ dài được xác định là 4, YEAR có thể từ 1901 tới 2155. Độ dài mặc định là 4.

## Kiểu dữ liệu chuỗi (String Data Types)

Dưới đây liệt kê các kiểu dữ liệu chuỗi và phần miêu tả của chúng trong MySQL:

Kiểu dữ liệu	Mô tả
CHAR(N)	Một chuỗi có độ dài cố định trong khoảng từ 1 đến 255 ký tự (ví dụ CHAR (5)). Nếu giá trị thật của một trường kiểu CHAR không bằng với độ dài khai báo thì phần thiếu bên phải của nó sẽ được thêm bằng các ký tự trắng một cách tự động. Việc xác định độ dài là không bắt buộc, nhưng mặc định là 1.
VARCHAR(N)	Một chuỗi có độ dài thay đổi, có độ dài từ 1 đến 255 ký tự (ví dụ VARCHAR(20)). Bạn phải định nghĩa độ dài khi tạo trường VARCHAR.
BLOB TEXT	Độ dài tối đa 65.535 ký tự. BLOB (Binary Large Objects) dùng để lưu trữ một lượng lớn dữ liệu nhị phân như các bức ảnh hoặc các loại tập tin khác. TEXT cũng lưu trữ được một lượng lớn dữ liệu. Sự khác biệt giữa hai loại trên là dữ liệu được lưu trữ phân biệt chữ hoa chữ thường trên các BLOB và không phân biệt chữ hoa chữ thường trong các trường TEXT. Không cần phải xác định độ dài với BLOB hoặc TEXT.
ENUM	Tạo danh sách các mục mà từ đó giá trị phải được chọn (hoặc nó có thể là NULL). Ví dụ, nếu ta muốn một trường nào đó chỉ nhận một trong các giá trị 'A' hoặc 'B' hoặc 'C' thì ta phải định nghĩa kiểu ENUM cho nó như sau: ENUM ('A', 'B', 'C'). Và chỉ có các giá trị này (hoặc NULL) có thể xuất hiện trong trường đó.

# CHƯƠNG 5. CÁC LỆNH SQL CƠ BẢN

## 5.1 Chú thích trong SQL

Trong các hệ quản trị cơ sở dữ liệu, chú thích là những đoạn văn bản sẽ bị bỏ qua khi chúng ta thực thi chương trình.

### Chú thích trên một dòng

Để viết chú thích trên một dòng thì ta đặt nội dung chú thích phía sau dấu -- hoặc dấu #



Ví dụ,

```
--Cú pháp dùng để tạo mới một database  
CREATE DATABASE database_name;
```

### Chú thích trên nhiều dòng

Để viết chú thích trên nhiều dòng thì ta đặt nội dung chú thích bên trong cặp dấu /\* và \*/



Ví dụ,

```
/* Câu truy van này dùng  
hien thi thong tin MSSV  
co trong bang SINHVIEN  
*/  
SELECT MSSV FROM SINHVIEN;
```

### Chú thích ở giữa dòng

Để chú thích ở giữa dòng, chúng ta đặt nội dung chú thích nằm bên trong cặp /\* và \*/



Ví dụ,

```
SELECT /* Tat ca cac cot*/ *  
FROM SINHVIEN;
```

## 5.2 Tạo Database

Câu lệnh CREATE DATABASE được sử dụng để tạo một cơ sở dữ liệu mới. Cần phải lưu ý rằng mỗi Database chỉ tồn tại với một tên duy nhất, không được phép trùng tên với các Database sẵn có.

### Cú pháp tạo Database

```
CREATE DATABASE [IF NOT EXISTS] database_name
[CHARACTER SET 'charset_name' COLLATE 'collate_name'];
```

*Trong đó:*

- `database_name` là tên database muốn tạo
- `IF NOT EXISTS` dùng để kiểm tra tên Database có tồn tại trước đó hay không



Thông thường khi tạo một Database trong MySQL chúng ta sẽ thiết lập CHARACTER SET là utf8mb4 và COLLATE là utf8mb4\_unicode\_ci để khi nhập tiếng Việt không bị lỗi font.

Để hiển thị danh sách Database đang tồn tại trong MySQL, ta sử dụng lệnh SHOW.

### Cú pháp hiển thị danh sách database

```
SHOW DATABASES;
```



Để chỉ định sử dụng một Database nào đó trong danh sách Database, ta sử dụng lệnh USE.

### Cú pháp chỉ định sử dụng database

```
USE database_name;
```



Ví dụ: Tạo một cơ sở dữ liệu mới tên là <testDB>, thì câu lệnh CREATE DATABASE sẽ như sau

```
CREATE DATABASE [IF NOT EXISTS] testDB;
```

Câu lệnh hiển thị tất cả các cơ sở dữ liệu hiện có

```
SHOW DATABASES;
```

Câu lệnh chỉ định sử dụng database testDB

```
USE testDB;
```

## 5.3 Tạo Bảng

Câu lệnh CREATE TABLE được sử dụng để tạo bảng trong một cơ sở dữ liệu. Mỗi bảng chỉ tồn tại với một tên duy nhất, không trùng lặp với tên các bảng sẵn có trong Database. Một bảng phải bao gồm danh sách các cột (column), danh sách các kiểu dữ liệu (data type) và các ràng buộc toàn vẹn (integrity constraint) trên từng cột nếu có.

### Cú pháp tạo bảng

```
CREATE TABLE [IF NOT EXISTS] table_name (
    column1 data_type [integrity_constraint],
    column2 data_type [integrity_constraint]
    ...
);
```

Trong đó:

- *table\_name* là tên bảng muốn tạo
- *column1, column2, ...* là tên cột thứ nhất, thứ hai,... của bảng
- *data\_type* là kiểu dữ liệu của cột
- *integrity\_constraint* là ràng buộc dữ liệu (nếu có) trên cột.



Ví dụ: Câu lệnh sau tạo ra một bảng SINHVIEN với cột MSSV là cột khóa chính và NOT NULL là những ràng buộc cho thấy các trường này không thể là NULL trong khi tạo các bản ghi trong bảng này.

```
CREATE TABLE SINHVIEN (
    MSSV INT PRIMARY KEY,
    TEN VARCHAR (20) NOT NULL,
    TUOI INT NOT NULL,
    DIACHI VARCHAR (30)
);
```

Ví dụ: Đoạn mã sau dùng để tạo các bảng MONTHI, KETQUA.

```
CREATE TABLE MONTHI (
    MAMT VARCHAR(4) PRIMARY KEY,
    TENMT VARCHAR (50) NOT NULL
);

CREATE TABLE KETQUA (
    MSSV INT NOT NULL,
    MAMT VARCHAR(4) NOT NULL,
    DIEM DECIMAL(3,1)
);
```

## 5.4 Câu lệnh INSERT INTO

Câu lệnh INSERT INTO được sử dụng để thêm dữ liệu vào một bảng. Trong nhiều trường hợp chúng ta có thể không cần phải chỉ rõ tên cột trong câu truy vấn nếu như ta đang thêm các giá trị cho tất cả các cột của bảng. Nhưng chúng ta cần phải chắc chắn rằng thứ tự của các giá trị là đúng theo thứ tự như các cột trong bảng.

### Cú pháp thêm dữ liệu vào bảng

```
INSERT INTO table_name (column1, column2,...) VALUES (value1,value2,...);
hoặc
INSERT INTO table_name VALUES (value1,value2,...);
```

Trong đó:

- column1, column2,... là tên cột thứ nhất, thứ hai,... của bảng
- value1, value2,... lần lượt là giá trị của cột 1, cột 2,...



Ví dụ: Thêm dữ liệu vào bảng SINHVIEN.

```
INSERT INTO SINHVIEN (MSSV, TEN, TUOI, DIACHI) VALUES (12150,'Doan Du',28,'HCM');
INSERT INTO SINHVIEN VALUES(12151,'Vuong Ngu Yen',25,'Long An');
INSERT INTO SINHVIEN VALUES
(12152,'Nham Doanh Doanh',23,'Tay Ninh'),
(12153,'Ly mac sau',23,'Binh Duong'),
(12154,'Au Duong Phong',23,'HCM');
```

Có thể sử dụng lệnh INSERT INTO để thêm dữ liệu vào một bảng thông qua câu lệnh SELECT trên một bảng khác. Cú pháp này thường được sử dụng trong những trường hợp chúng ta muốn sao chép dữ liệu từ một số cột nào đó của bảng table1 vào một số cột nào đó của bảng table2.

### Cú pháp câu lệnh INSERT INTO

```
INSERT INTO table2 [(column1, column2, column3,...)]
SELECT column1, column2, column3, ...
FROM table1 [WHERE condition];
```



Ví dụ: Sao chép dữ liệu của hai cột MSSV, TEN trong bảng SINHVIEN vào hai cột MSSV, HOTEN của bảng SINHVIEN1.

```
INSERT INTO SINHVIEN1(MSSV, HOTEN)
SELECT MSSV, TEN FROM SINHVIEN;
```

## 5.5 Câu lệnh ALTER

Câu lệnh ALTER TABLE được sử dụng để sửa đổi cấu trúc của một bảng.

### Cú pháp câu lệnh ALTER

- Thêm một cột mới

```
ALTER TABLE table_name ADD COLUMN column_name data_type;
```

- Xóa một cột trong bảng

```
ALTER TABLE table_name DROP COLUMN column_name;
```

- Sửa kiểu dữ liệu của cột

```
ALTER TABLE table_name [MODIFY/ALTER] COLUMN column_name data_type;
```

- Đổi tên bảng

```
ALTER TABLE table_name RENAME TO new_table_name;
```

- Thêm ràng buộc vào bảng

```
ALTER TABLE table_name ADD CONSTRAINT constraint_name constraint_type;
```

- Xóa ràng buộc ra khỏi bảng

```
ALTER TABLE table_name DROP CONSTRAINT constraint_name;
```



Sau đây là một số ví dụ sử dụng câu lệnh ALTER TABLE:

Thêm cột giới tính vào bảng SINHVIEN.

```
ALTER TABLE SINHVIEN ADD COLUMN GIOITINH VARCHAR(3);
```

Xóa cột tuổi ra khỏi bảng SINHVIEN.

```
ALTER TABLE SINHVIEN DROP COLUMN TUOI;
```

Để thay đổi kiểu dữ liệu cột DIACHI thành VARCHAR(100) trong MySQL ta dùng lệnh:

```
ALTER TABLE SINHVIEN MODIFY COLUMN DIACHI VARCHAR(100);
```

Thêm ràng buộc CHECK cho cột giới tính.

```
ALTER TABLE SINHVIEN ADD CONSTRAINT check_GIOITINH CHECK (GIOITINH IN ('Nam', 'Nu'));
```

Thêm ràng buộc NOT NULL cho cột địa chỉ.

```
ALTER TABLE SINHVIEN MODIFY COLUMN DIACHI VARCHAR(100) NOT NULL;
```

Xoá ràng buộc CHECK giới tính.

```
ALTER TABLE SINHVIEN DROP CONSTRAINT check_GIOITINH;
```

## 5.6 Câu lệnh DROP

Câu lệnh `DROP DATABASE` được sử dụng để xóa một Database, câu lệnh `DROP TABLE` được sử dụng để xóa một bảng ra khỏi một Database.

### Cú pháp xóa database

```
DROP DATABASE [IF EXISTS] database_name;
```



Ví dụ: Xóa Database `<testDB>` ra khỏi hệ thống.

```
/* Xóa Database testDB */
DROP DATABASE [IF EXISTS] testDB;
```

Lưu ý: Việc xóa một Database sẽ xóa hết tất cả những bảng nằm bên trong Database đó. Vì vậy, cần phải cân nhắc thật kỹ trước khi thực hiện câu lệnh này.

### Cú pháp xóa bảng

```
DROP TABLE table_name;
```



Ví dụ: Xóa bảng `SINHVIEN` ra khỏi Database

```
DROP TABLE SINHVIEN;
```

Lưu ý: Việc xóa một bảng sẽ xóa hết tất cả những dữ liệu nằm ở bên trong bảng và đồng thời cũng xóa bảng đó ra khỏi Database. Vì vậy, bạn cũng cần phải cân nhắc thật kỹ trước khi thực hiện câu lệnh này.

## 5.7 Câu lệnh TRUNCATE

Đối với trường hợp ta chỉ muốn xóa dữ liệu bên trong bảng, không xóa bảng thì ta có thể thay thế lệnh `DROP TABLE` bằng việc sử dụng lệnh `TRUNCATE TABLE`.

### Cú pháp câu lệnh TRUNCATE

```
TRUNCATE TABLE table_name;
```



Ví dụ: Xóa hết tất cả dữ liệu bên trong bảng `SINHVIEN`

```
TRUNCATE TABLE SINHVIEN;
```

## 5.8 Câu lệnh SELECT

Trong các hệ quản trị cơ sở dữ liệu, lệnh SELECT được dùng để truy xuất dữ liệu từ cơ sở dữ liệu. Để sử dụng lệnh SELECT thì chúng ta thường dùng 2 cú pháp như sau:

### Cú pháp 1

```
SELECT column1, column2, column3, ...
FROM name_table;
```

*Trong đó*

- *table\_name là tên của bảng mà chúng ta muốn lấy dữ liệu.*
- *column1, column2, ... là tên của những cột muốn truy xuất.*



Ví dụ: Lấy ra cột MSSV, TEN của bảng SINHVIEN

```
SELECT MSSV, TEN FROM SINHVIEN;
```

### Cú pháp 2

```
SELECT * FROM name_table;
```

*Trong đó \* dùng để lấy hết tất cả các cột bên trong bảng.*



Ví dụ: Lấy ra thông tin tất cả các cột của bảng SINHVIEN

```
SELECT * FROM SINHVIEN;
```

Trong một bảng đôi khi một cột thường có thể có nhiều giá trị trùng lặp, nhưng nếu chúng ta chỉ muốn lấy các giá trị khác nhau thì ta sử dụng lệnh SELECT DISTINCT. Chức năng của lệnh này dùng để truy xuất dữ liệu với mỗi loại giá trị chỉ lấy một lần duy nhất.

### Cú pháp DISTINCT

```
SELECT DISTINCT column1, column2, column3, ...
FROM table_name;
```



Ví dụ: Lấy ra danh sách tuổi của sinh viên và loại bỏ các giá trị trùng lặp.

```
SELECT DISTINCT TUOI
FROM SINHVIEN;
```

## 5.9 Mệnh đề WHERE

Thông thường, khi chúng ta sử dụng lệnh SELECT để truy xuất dữ liệu từ một bảng thì mặc định nó sẽ lấy hết tất cả các hàng nằm trên bảng đó. Tuy nhiên, trong nhiều trường hợp chúng ta chỉ muốn lấy những hàng khi dữ liệu trên hàng đó thỏa một điều kiện cụ thể nào đó.

Để giải quyết vấn đề được đặt ra ở phía trên thì khi sử dụng lệnh SELECT, chúng ta phải thêm mệnh đề WHERE nằm cuối câu lệnh với cú pháp như sau:

### Cú pháp SELECT...FROM...WHERE

```
SELECT column1, column2, column3, ...
FROM table_name
[WHERE condition];
```

*Trong đó*

- *condition là biểu thức điều kiện mà ta muốn dựa vào nó để chọn lọc dữ liệu, một biểu thức điều kiện thường có ba thành phần cơ bản là: <tên cột> <toán tử so sánh> <giá trị>*.



Ví dụ: In ra danh sách những sinh viên có tuổi lớn hơn hoặc bằng 20.

```
SELECT *
FROM SINHVIEN
WHERE TUOI >=20;
```

Mệnh đề WHERE ngoài việc được sử dụng bởi lệnh SELECT thì nó còn được dùng bởi lệnh UPDATE để cập nhật dữ liệu.

### Cú pháp UPDATE...WHERE...

```
UPDATE table_name
SET column1 = value1, column2 = value2..., columnN = valueN
[WHERE condition];
```



hoặc được dùng bởi lệnh DELETE để xóa dữ liệu.

### Cú pháp DELETE...WHERE...

```
DELETE FROM table_name
[WHERE condition];
```



## 5.10 Toán tử AND, OR, NOT

Toán tử AND, OR và NOT thường được sử dụng kết hợp với mệnh đề WHERE để cụ thể hóa việc xác định những dòng mà chúng ta muốn chọn lọc để thực hiện các thao tác truy vấn dữ liệu. Trong đó:

- Toán tử AND cho phép sự tồn tại của nhiều điều kiện trong mệnh đề WHERE.
- Toán tử OR lấy những hàng mà dữ liệu trên hàng đó thỏa ít nhất một trong số các điều kiện được đặt ra.
- Toán tử NOT lấy những hàng mà dữ liệu trên hàng đó không thỏa điều kiện được đặt ra.

### Cú pháp AND

```
SELECT column1, column2, column3, ...
FROM table_name
WHERE condition1 AND condition2 AND condition3...;
```



### Cú pháp OR

```
SELECT column1, column2, column3, ...
FROM table_name
WHERE condition1 OR condition2 OR condition3...;
```



### Cú pháp NOT

```
SELECT column1, column2, column3, ...
FROM table_name
WHERE NOT condition;
```



Ví dụ: In ra danh sách các sinh viên nữ và có tuổi lớn hơn hoặc bằng 18.

```
SELECT * FROM SINHVIEN WHERE GIOITINH = 'Nu' AND TUOI >=18;
```

Ví dụ: In ra danh sách các sinh viên sống tại HCM hoặc có tuổi lớn hơn hoặc bằng 18.

```
SELECT * FROM SINHVIEN WHERE DIACHI = 'HCM' OR TUOI >=18;
```

Ví dụ: In ra danh sách các sinh viên không sống tại HCM.

```
SELECT * FROM SINHVIEN WHERE NOT DIACHI = 'HCM';
```

## 5.11 Toán tử IN

Toán tử IN thường được sử dụng bởi mệnh đề WHERE để thực hiện việc chọn lọc dữ liệu theo điều kiện. Toán tử IN sẽ khai báo một tập hợp các giá trị, nếu giá trị dữ liệu của cột cần tìm trùng khớp với một trong số những giá trị được khai báo thì tức là thỏa điều kiện.

Toán tử IN thường có hai cách sử dụng là: khai báo một tập hợp những giá trị cụ thể hoặc khai báo một câu lệnh chọn lọc dữ liệu để trả về một cột các giá trị.

### Cú pháp 1

```
SELECT column1, column2, column3, ...
FROM table_name
WHERE column_name IN (value1, value2, ...);
```



Ví dụ: In ra danh sách các sinh viên có tuổi là 18 và 20.

```
SELECT * FROM SINHVIEN
WHERE TUOI IN (18, 20);
```

### Cú pháp 2

```
SELECT column1, column2, column3, ...
FROM table_name
WHERE column_name IN (<câu lệnh SELECT>);
```



Ví dụ: Liệt kê những sinh viên có ít nhất một môn thi trên 8 điểm.

```
SELECT * FROM SINHVIEN
WHERE MSSV IN (SELECT DISTINCT MSSV FROM KETQUA WHERE DIEM > 8);
```

Lưu ý: Toán tử IN

- Được sử dụng để giảm thiểu nhiều điều kiện OR.
- So sánh các giá trị giữa câu truy vấn con và câu truy vấn mẹ.
- Quét tất cả các giá trị bên trong khối IN.
- Có thể trả về TRUE, FALSE hoặc NULL. Do đó, chúng ta có thể sử dụng toán tử IN để so sánh các giá trị NULL.
- Có thể sử dụng trên các truy vấn con cũng như với các giá trị.
- Thực thi nhanh hơn khi kết quả truy vấn con ít hơn.

## 5.12 Toán tử EXISTS

Toán tử EXISTS được sử dụng để kiểm tra sự tồn tại của một bản ghi nào đó trong câu truy vấn. Việc kiểm tra bằng toán tử EXISTS sẽ trả về TRUE nếu truy vấn con trả về một hoặc nhiều bản ghi và ngược lại nếu toán tử EXISTS trả về FALSE thì nghĩa là không có bản ghi nào được trả về.

### Cú pháp

```
SELECT column1, column2, column3, ...
FROM table_name
WHERE [NOT] EXISTS (
    SELECT columns FROM table_name WHERE condition
);
```



Ví dụ: In ra danh sách những sinh viên có ít nhất một môn thi trên 9 điểm

```
SELECT *
FROM SINHVIEN
WHERE EXISTS (
    SELECT *
    FROM KETQUA
    WHERE (SINHVIEN.MSSV = KETQUA.MSSV) AND DIEM > 9);
```

Ví dụ: In ra danh sách những giáo viên không phải là giáo viên bộ môn CNTT

```
SELECT *
FROM GIAOVIEN
WHERE NOT EXISTS (
    SELECT *
    FROM BOMON
    WHERE (GIAOVIEN.MABM = BOMON.MABM) AND MABM = 'CNTT');
```

Lưu ý: Toán tử EXISTS

- Được sử dụng để kiểm tra sự tồn tại của dữ liệu trong một truy vấn con. Nói cách khác, nó xác định liệu giá trị có được trả về hay không.
- Không so sánh các giá trị giữa truy vấn con và truy vấn mẹ.
- Dùng để thực hiện thêm khi đáp ứng điều kiện tích cực duy nhất.
- Trả về TRUE hoặc FALSE. Do đó, không thể sử dụng nó để so sánh các giá trị NULL.
- Chỉ có thể sử dụng nó trên các truy vấn con.
- Thực thi nhanh hơn khi kết quả truy vấn con lớn.

## 5.13 Toán tử BETWEEN

Toán tử BETWEEN được sử dụng bởi mệnh đề WHERE để thực hiện việc chọn lọc dữ liệu theo điều kiện. Toán tử BETWEEN sẽ xác định một phạm vi giá trị (có thể là số, văn bản, ngày tháng), nếu giá trị dữ liệu của cột cần tìm nằm trong phạm vi này thì tức là thỏa điều kiện.

Nếu muốn xác định dữ liệu trong ngoài khoảng nào đó ta sử dụng NOT BETWEEN.

### Cú pháp

```
SELECT column1, column2, column3, ...
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```



Ví dụ: Liệt kê những sinh viên có tuổi từ 20 đến 25.

```
SELECT *
FROM SINVIEN
WHERE TUOI BETWEEN 20 AND 25;
```

Ví dụ: Lấy danh sách sinh viên có tên gồm các ký tự trong khoảng “A” đến “B”.

```
SELECT *
FROM SINVIEN
WHERE TEN BETWEEN 'A' AND 'B';
```

Ví dụ: Liệt kê những sinh viên được sinh ra trong khoảng thời gian từ 01/01/1998 đến 31/12/2003.

```
SELECT *
FROM SINVIEN
WHERE NGAYSINH BETWEEN '1998-01-01' AND '2003-12-31';
```

Ví dụ: Liệt kê danh sách sinh viên có năm sinh không nằm trong khoảng 1998 và 2003.

```
SELECT *
FROM SINVIEN
WHERE YEAR(NGAYSINH) NOT BETWEEN 1998 AND 2003;
```

## 5.14 Toán tử UNION

Toán tử UNION và UNION ALL được dùng để gộp tập kết quả của hai hay nhiều câu lệnh truy vấn lại với nhau. Trong đó:

- Toán tử UNION chỉ lấy mỗi loại giá trị một lần duy nhất (tức là không có trường hợp các giá trị trùng nhau)
- Toán tử UNION ALL sẽ lấy hết tất cả các kết quả, mặc cho chúng có bị trùng nhau hay không.

Khi sử dụng toán tử UNION hoặc UNION ALL thì chúng ta cần phải tuân thủ các quy tắc sau:

- Các câu lệnh SELECT phải có chung số lượng cột.
- Thứ tự cột trong các câu lệnh SELECT phải được sắp xếp theo đúng thứ tự cột của bảng kết hợp.
- Các cột tương ứng giữa các câu lệnh SELECT phải có kiểu dữ liệu giống nhau.

### Cú pháp UNION

```
SELECT column1, column2, column3, ... FROM table1 [WHERE condition]
UNION
SELECT column1, column2, column3, ... FROM table2 [WHERE condition];
```



### Cú pháp UNION ALL

```
SELECT column1, column2, column3, ... FROM table1 [WHERE condition]
UNION ALL
SELECT column1, column2, column3, ... FROM table2 [WHERE condition];
```



Ví dụ: Gộp các giáo viên và sinh viên (có cùng họ Nham) vào chung một bảng. Chỉ lấy các thông tin họ tên và tuổi.

```
SELECT TEN, TUOI FROM SINHVIEN WHERE TEN LIKE 'Nham%'
UNION ALL
SELECT HOTEN, TUOI FROM GIAOVIEN WHERE HOTEN LIKE 'Nham%'
```

Khi gộp kết quả từ hai hay nhiều câu lệnh truy vấn dữ liệu (SELECT) lại với nhau thì mặc định tên cột trên bảng kết quả sẽ được lấy từ tên của các cột ở trong câu lệnh truy vấn thứ nhất (điển hình như trong ví dụ, tên của các cột là TEN, TUOI chứ không phải là HOTEN, TUOI). Ta có thể sử dụng lệnh AS để thay đổi tên cột trên bảng kết hợp.

## 5.15 Câu lệnh ANY, ALL

Lệnh ANY và ALL thường được sử dụng kết hợp với các cột và các toán tử để xây dựng nên một biểu thức, biểu thức này thường gồm có bốn thành phần chính: <tên cột> <toán tử so sánh> <từ khóa ANY hoặc ALL> và <một câu lệnh SELECT> để trả về một cột dữ liệu.

### Cú pháp ANY

```
<column> <operator> ANY (<câu lệnh SELECT>)
```



Biểu thức này sẽ trả về giá trị TRUE nếu cột column có thể kết hợp với ít nhất một giá trị bên trong cột dữ liệu (được trả về từ câu lệnh SELECT) làm cho biểu thức <column> <operator> <value> thỏa điều kiện.

### Cú pháp ALL

```
<column> <operator> ALL (<câu lệnh SELECT>)
```



Biểu thức này sẽ trả về giá trị TRUE nếu tất cả các lần kết hợp giữa cột column và các giá trị bên trong cột dữ liệu (được trả về từ câu lệnh SELECT) đều làm cho biểu thức <column> <operator> <value> thỏa điều kiện.

Lưu ý: <operator> là các toán tử so sánh như: = , > , >= , < , <= , != , <>

Ví dụ: In ra danh sách những sinh viên có tuổi lớn hơn bất kỳ một sinh viên nào sống tại HCM.

```
SELECT * FROM SINHVIEN
WHERE TUOI > ANY (
    SELECT TUOI
    FROM SINHVIEN
    WHERE DIACHI = 'HCM'
);
```

Ví dụ: In ra danh sách những sinh viên có tuổi lớn hơn tất cả các sinh viên sống tại HCM.

```
SELECT * FROM SINHVIEN
WHERE TUOI > ALL (
    SELECT TUOI
    FROM SINHVIEN
    WHERE DIACHI = 'HCM'
);
```

## 5.16 Câu lệnh AS

Lệnh AS được dùng để đặt *bí danh* (tên tạm thời) cho các cột hoặc bảng, những bí danh này chỉ tồn tại trong khoảng thời gian truy vấn dữ liệu.

Việc đặt bí danh thường được sử dụng trong những trường hợp như:

- Khi truy xuất dữ liệu, bạn muốn đổi tên cho các cột trở nên ngắn gọn, dễ nhìn, dễ hiểu hơn,...
- Khi có nhiều bảng tham gia vào việc truy vấn dữ liệu, các chúng ta muốn đặt lại tên cho bảng ngắn hơn, để đơn giản hóa việc viết mã lệnh.

### Cú pháp đặt bí danh cho cột

```
SELECT column_name AS alias_name
FROM table_name;
```



### Cú pháp đặt bí danh cho bảng

```
SELECT column_name
FROM table_name AS alias_name;
```



Ví dụ: Liệt kê những sinh viên có độ tuổi từ 20 đến 25.

```
SELECT MSSV AS 'Mã sinh viên', ten AS 'Tên sinh viên'
FROM SINHVIEN
WHERE TUOI BETWEEN 20 AND 25;
```

Ví dụ: Liệt kê mã số sinh viên, họ tên và tên ngành thi của sinh viên sống tại HCM

```
SELECT SV.MSSV, SV.TEN, NG.TENNGANH
FROM SINHVIEN AS SV, NGANH AS NG
WHERE SV.MANGANH = NG.MANGANH AND DIACHI = 'HCM' ;
```

Câu lệnh phía trên sẽ có ý nghĩa tương tự như câu lệnh sau:

```
SELECT SINHVIEN.MSSV, SINHVIEN.TEN, NGANH.TENNGANH
FROM SINHVIEN, NGANH
WHERE SINHVIEN.MSSV = NGANH.MSSV AND DIACHI = 'HCM' ;
```

## 5.17 Câu lệnh WITH

Common Table Expression (CTE) là nơi lưu trữ bảng dữ liệu được truy xuất một cách tạm thời trong bộ nhớ dưới một cái tên để có thể sử dụng lại về sau.

Lệnh WITH chính là cú pháp để chúng ta sử dụng chức năng CTE, nó dùng để cung cấp một khối truy vấn con mà có thể được tham chiếu trong một số vị trí bên trong truy vấn chính.

### Cú pháp

```
WITH alias1_name AS (<Câu truy vấn con>),
      alias2_name AS (<Câu truy vấn con>),...
SELECT columns_name
FROM alias1_name [table_name]
[WHERE condition];
```



Ví dụ: In ra danh sách những sinh viên nam.

```
WITH SVNAM AS (
SELECT * FROM SINHVIEN WHERE GIOITINH = 'Nam'
)
SELECT MSSV, TEN FROM SVNAM;
```

Ví dụ: In ra danh sách sinh viên, điểm các môn thi và tổng điểm thi các môn của sinh viên đó.

```
WITH KQ AS (
SELECT * FROM KETQUA
),
TONGDIEMSV AS (
SELECT MSSV, SUM(DIEM) AS TONGDIEN
FROM KQ -- sử dụng lại bảng CTE KQ
GROUP BY MSSV
)
SELECT KQ.*, TONGDIEMSV.TONGDIEN -- sử dụng lại bảng CTE KQ
FROM KQ JOIN TONGDIEMSV USING (MSSV); -- join với bảng CTE TONGDIEMSV
```

Một số ưu điểm của lệnh WITH:

- Tối ưu hóa bộ nhớ và tốc độ.
- Giảm độ phức tạp của query.
- Dễ đọc và dễ hiểu.
- Tăng tính linh hoạt mà sub-query không có.
- Khả năng lồng ghép câu lệnh như Sub-query.

## 5.18 Câu lệnh CASE

Lệnh CASE được dùng để tạo ra những giá trị khác nhau dựa trên những điều kiện khác nhau.

### Cú pháp CASE

CASE

```
WHEN condition1 THEN result1;
WHEN condition2 THEN result2;
...
WHEN conditionN THEN resultN;
ELSE result;
END;
```



Lệnh CASE sẽ kiểm tra lần lượt các điều kiện (theo thứ tự từ trên xuống), nếu phát hiện được điều kiện đúng thì nó sẽ kết thúc việc kiểm tra và trả về giá trị nằm tại điều kiện đó.

- Nếu không có điều kiện nào đúng thì lệnh CASE sẽ trả về giá trị tại mệnh đề ELSE.
- Nếu không có điều kiện nào đúng và không có mệnh đề ELSE thì trả về giá trị NULL.

Ví dụ: Liệt kê tên, tuổi, nhóm tuổi của tất cả các sinh viên.

- Nếu sinh viên có tuổi lớn hơn 20 thì nhóm tuổi sẽ ghi là “Lớn hơn 20 tuổi”
- Nếu sinh viên có tuổi bằng 20 thì nhóm tuổi sẽ ghi là “Bằng 20 tuổi”
- Ngược lại sẽ ghi là “Nhỏ hơn 20 tuổi”

```
SELECT TEN, TUOI,
CASE
    WHEN TUOI > 20 THEN 'Lon hon 20 tuoi'
    WHEN TUOI = 20 THEN 'Bang 20 tuoi'
    ELSE 'Nho hon 20 tuoi'
END AS NHOMTUOI
FROM SINHVIEN;
```

Ví dụ: Liệt kê thông tin của tất cả các sinh viên. Nếu biểu thức  $(4 + 10) > 15$  là đúng thì sắp xếp thứ tự các sinh viên theo độ tuổi, còn nếu sai thì sắp xếp theo giới tính.

```
SELECT * FROM SINHVIEN
ORDER BY
(CASE
    WHEN (4+10) > 15 THEN TUOI
    ELSE GIOITINH
END);
```

## 5.19 Câu lệnh UPDATE

Câu lệnh UPDATE được sử dụng để sửa đổi các bản ghi hiện có trong một bảng. Chúng ta có thể sử dụng mệnh đề WHERE với truy vấn UPDATE để cập nhật các bản ghi đã chọn, nếu không tất cả các bản ghi sẽ bị ảnh hưởng.

### Cú pháp UPDATE

```
UPDATE table_name
SET column1 = value1,
    column2 = value2,
    ...
    columnN = valueN
[WHERE condition];
```



Ví dụ: Cập nhật điểm môn Toán cho sinh viên có mã số sinh viên là 12153 thành 9.0.

```
UPDATE KETQUA
SET DIEM = 9.0
WHERE MSSV = 12153 AND MAMT = 'Toan';
```

Ví dụ: Cập nhật ngày sinh của giáo viên “Lê Minh Anh” là 15/12/1990 và lương là 2000.

```
UPDATE GIAOVIEN
SET NGSINH = '1990-12-15', LUONG = 2000
WHERE HOTEN = 'Lê Minh Anh';
```

Khi sử dụng câu lệnh UPDATE, nếu chúng ta không xác định mệnh đề WHERE thì tất cả các dòng trong bảng đều sẽ bị cập nhật.

Ví dụ: Cập nhật tuổi cho tất cả sinh viên có trong bảng là 23.

```
UPDATE SINHVIEN
SET TUOI = 23;
```

## 5.20 Câu lệnh DELETE

Câu lệnh DELETE được sử dụng để xóa các bản ghi hiện có từ một bảng. Chúng ta có thể sử dụng mệnh đề WHERE với truy vấn DELETE để xóa các hàng đã chọn, nếu không tất cả các bản ghi sẽ bị xóa.

### Cú pháp DELETE

```
DELETE
FROM table_name
[WHERE conditions];
```



Ví dụ: Xóa sinh viên có mã số sinh viên là 12153 ra khỏi bảng.

```
DELETE
FROM SINHVIEN
WHERE MSSV = 12153;
```

Ví dụ: Xóa sinh toàn bộ sinh viên thuộc Khoa “Toán tin”.

```
DELETE
FROM SINHVIEN
WHERE MAKHOA IN (SELECT MAKHOA
                  FROM KHOA
                  WHERE TENKHOA = 'Toán tin');
```

Khi sử dụng lệnh DELETE, nếu chúng ta không khai báo mệnh đề WHERE để xác định những hàng nào sẽ được xóa thì mặc định tất cả các hàng bên trong bảng đều bị xóa.

Ví dụ: Câu truy vấn sau sẽ xoá hết tất cả các sinh viên có trong bảng.

```
DELETE FROM SINHVIEN;
```

## 5.21 Mệnh đề LIKE

Mệnh đề LIKE được sử dụng để so sánh một giá trị với các giá trị tương tự sử dụng toán tử ký tự đại diện (wildcard). Có hai ký tự đại diện được sử dụng kết hợp với toán tử LIKE:

- Phần trăm (%): Đại cho cho không hoặc nhiều ký tự bất kỳ.
- Dấu gạch dưới (\_): Đại diện cho một ký tự bất kỳ.

Mẫu chuỗi là một tập hợp gồm những ký tự bình thường được kết hợp với các ký tự đại diện để xây dựng nên một cái mô hình mô tả một dạng chuỗi ký tự nào đó. Ví dụ:

- %Comment chính là một mẫu chuỗi, nó dùng để mô tả cho những chuỗi ký tự được kết thúc bởi chữ Comment, còn phía trước chữ Comment thì có thể là không hoặc nhiều ký tự bất kỳ.
- \_Comment chính là một mẫu chuỗi, nó dùng để mô tả cho những chuỗi ký tự được kết thúc bởi chữ Comment, còn phía trước chữ Comment có một ký tự bất kỳ.

### Cú pháp LIKE

```
SELECT column1, column2, column3, ...
FROM table_name
WHERE columnN LIKE <mẫu chuỗi>;
```



Một vài ví dụ cho thấy mệnh đề WHERE có mệnh đề LIKE với toán tử '%' và '\_' khác nhau:

```
--Tìm danh sách các sinh viên có họ Quach
SELECT * FROM SINHVIEN WHERE TEN LIKE 'Quach%';

--Tìm danh sách các sinh viên sống tại HCM
SELECT * FROM SINHVIEN WHERE TEN LIKE '%HCM';

--Tìm danh sách các sinh viên có tên là Phuc hoặc tên lót là Hoang
SELECT * FROM SINHVIEN
WHERE (TEN LIKE '%Phuc') OR (ten LIKE '%_Hoang_%');

--Tìm danh sách các sinh viên không có tên là Nham
SELECT * FROM SINHVIEN WHERE HOTEN NOT LIKE '%Nham';
```

Khi khai báo một mẫu chuỗi, nếu trong số những ký tự mà chúng ta muốn tìm có chứa ký tự % hoặc ký tự \_ thì chúng ta cần phải thêm một dấu gạch chéo ngược phía trước những ký tự đó.

Ví dụ như mẫu chuỗi Nguyen%\\_ dùng để mô tả những chuỗi ký tự được bắt đầu bằng chữ Nguyen, đồng thời phải được kết thúc bởi một ký tự là dấu gạch dưới.

## 5.22 Mệnh đề ORDER BY

Mệnh đề ORDER BY dùng để sắp xếp dữ liệu theo thứ tự tăng dần hoặc giảm dần dựa trên một hoặc nhiều cột. Lệnh ASC được sử dụng để sắp xếp tăng dần và lệnh DESC được sử dụng để sắp xếp giảm dần.

Nếu chúng ta sắp xếp thứ tự của các kết quả trả về dựa trên nhiều cột thì những cột nào khai báo trước sẽ được sắp xếp trước.

### Cú pháp ORDER BY

```
SELECT column1, column2, column3, ...
FROM table_name
[WHERE conditions]
ORDER BY column1, column2, .. columnN [ASC or DESC];
```



Ví dụ: Cách sử dụng mệnh đề ORDER BY

```
--In ra danh sách sinh viên nam có tuổi tăng dần
SELECT * FROM SINHVIEN
WHERE GIOITINH = 'Nam'
ORDER BY TUOI ASC;

--In ra danh sách các bài thi có điểm giảm dần
SELECT * FROM KETQUA
ORDER BY DIEM DESC;

--In ra danh sách sinh viên có tuổi tăng dần và MSSV giảm dần
SELECT * FROM SINHVIEN
ORDER BY TUOI ASC, MSSV DESC;
```

Lưu ý: Khi sử dụng lệnh ORDER BY thì mặc định các kết quả trả về sẽ được sắp xếp theo thứ tự tăng dần. Cho nên, nếu muốn sắp xếp theo thứ tự tăng dần thì ta không cần phải thêm từ khóa ASC.

Ví dụ: Hai câu lệnh sau đều dùng để sắp xếp tuổi sinh viên tăng dần.

```
SELECT * FROM SINHVIEN
ORDER BY TUOI ASC;
```

là tương đương với

```
SELECT * FROM SINHVIEN
ORDER BY TUOI;
```

## 5.23 Mệnh đề LIMIT

Mệnh đề LIMIT dùng để hạn chế số lượng bản ghi được trả về bởi câu lệnh SELECT.

### Cú pháp

```
SELECT column1, column2, column3, ...
FROM table_name
[WHERE conditions]
LIMIT <số lượng dòng muốn lấy>;
```



Ví dụ: Câu truy vấn sau dùng để chọn 3 sinh viên đầu tiên từ bảng SINHVIEN.

```
SELECT *
FROM SINHVIEN
LIMIT 3;
```

Mệnh đề LIMIT thường được sử dụng với mệnh đề ORDER BY. Đầu tiên, chúng ta sử dụng mệnh đề ORDER BY để sắp xếp kết quả dựa trên các tiêu chí nhất định và sau đó sử dụng mệnh đề LIMIT để tìm giá trị thấp nhất hoặc cao nhất.

Ví dụ: In ra danh sách 3 sinh viên có tuổi nhỏ nhất

```
SELECT *
FROM SINHVIEN
ORDER BY TUOI ASC
LIMIT 3;
```

Ví dụ: In ra danh sách 3 bài thi có điểm thi cao nhất

```
SELECT *
FROM KETQUA
ORDER BY DIEM DESC
LIMIT 3;
```

Lưu ý: Không phải tất cả các hệ quản trị cơ sở dữ liệu đều hỗ trợ mệnh đề LIMIT. SQL Server hỗ trợ mệnh đề SELECT TOP, trong khi Oracle sử dụng ROWNUM để chọn một số lượng hạn chế các bản ghi.

Ví dụ:

```
SELECT TOP 3 * FROM KETQUA ORDER BY DIEM DESC; -- SQL Server

SELECT * FROM KETQUA ORDER BY DIEM DESC -- Oracle
WHERE ROWNUM <= 3;
```

## 5.24 Mệnh đề GROUP BY

Lệnh GROUP BY được dùng để nhóm những hàng có cùng giá trị dựa trên một cột nào đó lại với nhau, nó thường được sử dụng kết hợp với các hàm tính toán tổng hợp như MIN, MAX, SUM, COUNT, AVG,... để tính toán giá trị cột của những hàng được nhóm lại.

Mệnh đề GROUP BY tuân theo mệnh đề WHERE trong câu lệnh SELECT và thường đứng trước mệnh đề ORDER BY nếu mệnh đề được sử dụng.

### Cú pháp sử dụng GROUP BY

```
SELECT column_name(s)
FROM table_name
[WHERE conditions]
GROUP BY column_name [ORDER BY column_name];
```



Lưu ý: Cột hiển thị trong bảng kết quả sử dụng mệnh đề GROUP BY phải là thuộc tính của GROUP BY hoặc các hàm tính tổng hợp.

Ví dụ: Cho biết tổng số điểm của mỗi sinh viên.

```
SELECT MSSV, TEN, SUM(DIEM)
FROM SINHVIEN, KETQUA
WHERE SINHVIEN.MSSV = KETQUA.MSSV
GROUP BY MSSV, TEN;
```

Ví dụ: Cho biết tổng số bài thi trên 8 điểm của mỗi sinh viên.

```
SELECT SV.MSSV, SV.TEN, COUNT(SV.MSSV)
FROM SINHVIEN AS SV, KETQUA AS KQ
WHERE SV.MSSV = KQ.MSSV AND DIEM > 8
GROUP BY SV.MSSV, TEN;
```

Ví dụ: Xuất ra danh sách tên bộ môn và số lượng giáo viên của bộ môn đó.

```
SELECT TENBM, COUNT(*) AS 'So Luong'
FROM GIAOVIEN, BOMON
WHERE BOMON.MABM = GIAOVIEN.MABM
GROUP BY TENBM;
```

## 5.25 Mệnh đề HAVING

Mệnh đề HAVING được sử dụng thay thế cho mệnh đề WHERE khi điều kiện chọn lọc dữ liệu có nhắc đến các hàm tính toán tổng hợp như: MIN, MAX, SUM, COUNT, AVG,...

Mệnh đề WHERE đặt các điều kiện vào các cột đã chọn, trong khi mệnh đề HAVING đặt các điều kiện vào các nhóm được tạo bởi mệnh đề GROUP BY.

### Cú pháp

```
SELECT column_name(s)
FROM table_name
[WHERE conditions]
GROUP BY column_name
HAVING condition;
```



Ví dụ: In ra danh sách những sinh viên có tổng số điểm các môn thi > 28.

```
SELECT SV.MSSV, SV.TEN
FROM SINHVIEN AS SV, KETQUA AS KQ
WHERE SV.MSSV = KQ.MSSV
GROUP BY MSSV, TEN
HAVING SUM(DIEM) > 28;
```

Ví dụ: In ra danh sách những sinh viên có tất cả 3 bài thi đều trên 8 điểm.

```
SELECT SV.MSSV, SV.TEN
FROM SINHVIEN AS SV, KETQUA AS KQ
WHERE SV.MSSV = KQ.MSSV AND DIEM > 8
GROUP BY SV.MSSV, TEN
HAVING COUNT(SV.MSSV) = 3;
```

Ví dụ: Xuất ra số lượng giáo viên trong từng bộ môn có số giáo viên > 1.

```
SELECT TENBM, COUNT(*) AS SL
FROM GIAOVIEN, BOMON
WHERE BOMON.MABM = GIAOVIEN.MABM
GROUP BY TENBM
HAVING SL > 1;
```

## 5.26 Mệnh đề JOIN

Mệnh đề JOIN được sử dụng để kết hợp các dòng từ hai hay nhiều bảng trong cơ sở dữ liệu. JOIN là một phương tiện để kết hợp các cột từ hai bảng bằng cách sử dụng các giá trị chung cho mỗi bảng.

Chúng ta có thể sử dụng mệnh đề USING để join 2 bảng có cùng tên cột. USING sẽ tự động xóa cột bị trùng trong bảng kết quả.

Một số kiểu join thường được sử dụng như: INNER JOIN, LEFT JOIN, RIGHT JOIN,...

Ví dụ, chúng ta xét hai bảng sau:

**Bảng 1 - Bảng SINHVIEN**

MSSV	TEN	TUOI	DIACHI	GIOITINH
12150	Doan Du	28	HCM	Nam
12151	Vuong Ngu Yen	25	Long An	Nu
12152	Nham Doanh Doanh	26	Tay Ninh	Nu
12153	Ly Mac Sau	23	Binh Duong	Nu
12154	Au Duong Phong	23	HCM	Nam

**Bảng 2 - Bảng KETQUA**

MSSV	MAMT	DIEM
12150	CSDL	8.5
12150	THDC	9.0
12152	CSDL	9.0
12154	CSDL	9.5

Bây giờ, chúng ta sẽ join hai bảng bằng câu lệnh SELECT.

```
SELECT MSSV, TEN, MAMT, DIEM
FROM SINHVIEN, KETQUA
WHERE SINHVIEN.MSSV = KETQUA.MSSV;
```

hoặc

```
SELECT MSSV, TEN, MAMT, DIEM
FROM SINHVIEN JOIN KETQUA USING (MSSV);
```

Kết quả trả về là một bảng có nội dung như sau:

MSSV	TEN	MAMT	DIEM
12150	Doan Du	CSDL	8.5
12150	Doan Du	THDC	9.0
12152	Nham Doanh Doanh	CSDL	9.0
12154	Au Duong Phong	CSDL	9.5

## 5.27 Câu lệnh INNER JOIN

Lệnh INNER JOIN được sử dụng để kết hợp các hàng từ hai bảng lại thành một bảng tổng hợp, việc kết hợp các hàng sẽ được thực hiện dựa trên nguyên tắc như sau:

- Tất cả các hàng bên trong bảng thứ nhất sẽ được bắt cặp lần lượt với tất cả các hàng bên trong bảng thứ hai
- Những hàng mà dữ liệu trên nó thỏa điều kiện (condition) thì sẽ được giữ lại, còn những hàng mà dữ liệu trên nó không thỏa điều kiện thì sẽ bị loại bỏ.

Lưu ý: Khi hai bảng được kết hợp với nhau, các hàng trong bảng thứ nhất sẽ được đặt bên trái, các hàng trong bảng thứ hai sẽ được ghép bên phải.

### Cú pháp INNER JOIN

```
SELECT * FROM table1 [INNER] JOIN table2 ON condition;
```



Ví dụ, thực hiện câu lệnh INNER JOIN hai bảng SINHVIEN và KETQUA bằng đoạn mã bên dưới

```
SELECT *
FROM SINHVIEN
INNER JOIN KETQUA ON DIACHI = 'HCM' AND DIEM > 9;
```

Kết quả trả về là một bảng dữ liệu như sau:

MSSV	TEN	TUOI	DIACHI	GIOITINH	MSSV	MAMT	DIEM
12154	Au Duong Phong	23	HCM	Nam	12154	CSDL	9.5
12150	Doan Du	28	HCM	Nam	12154	CSDL	9.5

Ví dụ, thực hiện câu lệnh INNER JOIN hai bảng SINHVIEN và KETQUA bằng đoạn mã bên dưới

```
SELECT SINHVIEN.MSSV, TEN, MAMT, DIEM
FROM SINHVIEN
INNER JOIN KETQUA ON SINHVIEN.MSSV = KETQUA.MSSV;
```

Kết quả trả về là một bảng dữ liệu như sau:

MSSV	TEN	MAMT	DIEM
12150	Doan Du	CSDL	8.5
12150	Doan Du	THDC	9.0
12152	Nham Doanh Doanh	CSDL	9.0
12154	Au Duong Phong	CSDL	9.5

## 5.28 Câu lệnh LEFT JOIN

Lệnh LEFT JOIN được sử dụng để kết hợp các hàng từ hai bảng lại thành một bảng tổng hợp, việc kết hợp các hàng sẽ được thực hiện dựa trên nguyên tắc như sau:

- Tất cả các hàng bên trong bảng thứ nhất sẽ được bắt cặp lần lượt với tất cả các hàng bên trong bảng thứ hai
- Những hàng mà dữ liệu trên nó thỏa điều kiện (condition) thì sẽ được giữ lại, còn những hàng mà dữ liệu trên nó không thỏa điều kiện thì sẽ bị loại bỏ.
- Những hàng trong bảng thứ nhất không bắt cặp được với bất kỳ hàng nào bên trong bảng thứ hai mà thỏa điều kiện thì nó sẽ được ghép với một hàng chứa toàn giá trị NULL.

Lưu ý: Khi hai bảng được kết hợp với nhau, các hàng trong bảng thứ nhất sẽ được đặt bên trái, các hàng trong bảng thứ hai sẽ được ghép bên phải.

### Cú pháp LEFT JOIN

```
SELECT * FROM table1 LEFT JOIN table2 ON condition;
```



Ví dụ, thực hiện câu lệnh LEFT JOIN hai bảng SINHVIEN và KETQUA bằng đoạn mã bên dưới

```
SELECT *
FROM SINHVIEN
LEFT JOIN KETQUA
ON DIACHI = 'HCM' AND DIEM > 9;
```

Kết quả trả về là một bảng dữ liệu như sau:

## 5.28 Câu lệnh LEFT JOIN

MSSV	TEN	TUOI	DIACHI	GIOITINH	MSSV	MAMT	DIEM
12150	Doan Du	28	HCM	Nam	12154	CSDL	9.5
12151	Vuong Ngu Yen	25	Long An	Nu	NULL	NULL	NULL
12152	Nham Doanh Doanh	26	Tay Ninh	Nu	NULL	NULL	NULL
12153	Ly Mac Sau	23	Binh Duong	Nu	NULL	NULL	NULL
12154	Au Duong Phong	23	HCM	Nam	12154	CSDL	9.5

Ví dụ, thực hiện câu lệnh LEFT JOIN hai bảng SINHVIEN và KETQUA bằng đoạn mã bên dưới

```
SELECT *
FROM SINHVIEN
LEFT JOIN KETQUA
ON DIEM = 10;
```

Kết quả trả về là một bảng dữ liệu như sau:

MSSV	TEN	TUOI	DIACHI	GIOITINH	MSSV	MAMT	DIEM
12150	Doan Du	28	HCM	Nam	NULL	NULL	NULL
12151	Vuong Ngu Yen	25	Long An	Nu	NULL	NULL	NULL
12152	Nham Doanh Doanh	26	Tay Ninh	Nu	NULL	NULL	NULL
12153	Ly Mac Sau	23	Binh Duong	Nu	NULL	NULL	NULL
12154	Au Duong Phong	23	HCM	Nam	NULL	NULL	NULL

Ví dụ, thực hiện câu lệnh LEFT JOIN hai bảng SINHVIEN và KETQUA bằng đoạn mã bên dưới

```
SELECT SINHVIEN.MSSV, TEN, MAMT, DIEM
FROM SINHVIEN
LEFT JOIN KETQUA
ON SINHVIEN.MSSV = KETQUA.MSSV
ORDER BY DIEM DESC, MSSV DESC;
```

Kết quả trả về là một bảng dữ liệu như sau:

MSSV	TEN	MAMT	DIEM
12154	Au Duong Phong	CSDL	9.5
12152	Nham Doanh Doanh	CSDL	9.0
12150	Doan Du	THDC	9.0
12150	Doan Du	CSDL	8.5
12153	Ly Mac Sau	NULL	NULL
12151	Vuong Ngu Yen	NULL	NULL

## 5.29 Câu lệnh RIGHT JOIN

Lệnh RIGHT JOIN được sử dụng để kết hợp các hàng từ hai bảng lại thành một bảng tổng hợp, việc kết hợp các hàng sẽ được thực hiện dựa trên nguyên tắc như sau:

- Tất cả các hàng bên trong bảng thứ hai sẽ được bắt cặp lần lượt với tất cả các hàng bên trong bảng thứ nhất
- Những hàng mà dữ liệu trên nó thỏa điều kiện (condition) thì sẽ được giữ lại, còn những hàng mà dữ liệu trên nó không thỏa điều kiện thì sẽ bị loại bỏ.
- Những hàng trong bảng thứ hai không bắt cặp được với bất kỳ hàng nào bên trong bảng thứ nhất mà thỏa điều kiện thì nó sẽ được ghép với một hàng chứa toàn giá trị NULL.

Lưu ý: Khi hai bảng được kết hợp với nhau, các hàng trong bảng thứ nhất sẽ được đặt bên trái, các hàng trong bảng thứ hai sẽ được ghép bên phải.

### Cú pháp RIGHT JOIN

```
SELECT * FROM table1 RIGHT JOIN table2 ON condition;
```



Ví dụ, thực hiện câu lệnh RIGHT JOIN hai bảng SINHVIEN và KETQUA bằng đoạn mã bên dưới

```
SELECT *
FROM SINHVIEN
RIGHT JOIN KETQUA
ON DIACHI = 'HCM' AND DIEM > 9;
```

Kết quả trả về là một bảng dữ liệu như sau:

MSSV	TEN	TUOI	DIACHI	GIOITINH	MSSV	MAMT	DIEM
NULL	NULL	NULL	NULL	NULL	12150	CSDL	8.5
NULL	NULL	NULL	NULL	NULL	12150	THDC	9.0
NULL	NULL	NUL	NULL	NULL	12152	CSDL	9.0
12154	Au Duong Phong	21	HCM	Nam	12154	CSDL	9.5
12150	Doan Du	23	HCM	Nam	12154	CSDL	9.5

Ví dụ, thực hiện câu lệnh RIGHT JOIN hai bảng SINHVIEN và KETQUA bằng đoạn mã bên dưới

```
SELECT * FROM SINHVIEN
RIGHT JOIN KETQUA ON DIEM = 10;
```

Kết quả trả về là một bảng dữ liệu như sau:

MSSV	TEN	TUOI	DIACHI	GIOITINH	MSSV	MAMT	DIEM
NULL	NULL	NULL	NULL	NULL	12150	CSDL	8.5
NULL	NULL	NULL	NULL	NULL	12150	THDC	9.0
NULL	NULL	NULL	NULL	NULL	12152	CSDL	9.0
NULL	NULL	NULL	NULL	NULL	12154	CSDL	9.5

Ví dụ, thực hiện câu lệnh RIGHT JOIN hai bảng SINHVIEN và KETQUA bằng đoạn mã bên dưới

```
SELECT SINHVIEN.MSSV, TEN, MAMT, DIEM FROM SINHVIEN
RIGHT JOIN KETQUA ON SINHVIEN.MSSV = KETQUA.MSSV
ORDER BY DIEM DESC, MSSV DESC;
```

Kết quả trả về là một bảng dữ liệu như sau:

MSSV	TEN	MAMT	DIEM
12154	Au Duong Phong	CSDL	9.5
12152	Nham Doanh Doanh	CSDL	9.0
12150	Doan Du	THDC	9.0
12150	Doan Du	CSDL	8.5

## 5.30 Câu truy vấn con

Truy vấn con (truy vấn lồng) hay còn gọi là Sub Query, tức là một truy vấn sẽ được lồng trong một câu truy vấn khác. Bản chất của câu truy vấn con là trả về một bảng ảo, sau đó sử dụng bảng ảo đó để thực hiện tiếp các câu lệnh khác. Lệnh truy vấn con sẽ có tốc độ chậm hơn bình thường vì nó là tạo ra table ảo nên sẽ tốn nhiều tài nguyên lưu trữ hơn.

Ví dụ: In ra danh sách sinh viên có nhiều hơn 1 kết quả thi.

```
SELECT *
FROM SINHVIEN
WHERE ( SELECT COUNT(*)
        FROM KETQUA
        WHERE SINHVIEN.MSSV = KETQUA.MSSV ) > 1;
```

Ví dụ: In ra danh sách những sinh viên có tuổi lớn hơn tất cả các sinh viên sống tại HCM.

```
SELECT *
FROM SINHVIEN
WHERE TUOI > ALL (SELECT TUOI FROM SINHVIEN WHERE DIACHI = 'HCM');
```

Ví dụ: In ra danh sách sinh viên có tuổi > tuổi trung bình của tất cả các sinh viên

```
SELECT *
FROM SINHVIEN
WHERE TUOI > ((SELECT SUM(TUOI) FROM SINHVIEN) / (SELECT COUNT(*) FROM SINHVIEN));
```

Ví dụ: In ra danh sách sinh viên có điểm thi môn CSDL là cao nhất.

```
SELECT SV.MSSV, TEN
FROM SINHVIEN AS SV JOIN KETQUA AS KQ ON SV.MSSV = KQ.MSSV
WHERE DIEM = (SELECT MAX(DIEM) FROM KETQUA) AND MAMT = 'CSDL';
```

Ví dụ: In ra danh sách sinh viên có tổng điểm các môn thi là cao nhất.

```
SELECT SINHVIEN.MSSV, TEN
FROM SINHVIEN JOIN KETQUA ON SINHVIEN.MSSV = KETQUA.MSSV
GROUP BY SINHVIEN.MSSV
HAVING SUM(DIEM) >= ALL(
    SELECT SUM(DIEM)
    FROM SINHVIEN JOIN KETQUA ON SINHVIEN.MSSV = KETQUA.MSSV
    GROUP BY KETQUA.MSSV
);
```

# CHƯƠNG 6. CÁC LOẠI RÀNG BUỘC

Trong các hệ quản trị cơ sở dữ liệu, các ràng buộc (constraint) được sử dụng để xác định những quy tắc cho dữ liệu bên trong một bảng, nó sẽ giới hạn loại dữ liệu có thể thêm vào bên trong bảng. Điều này đảm bảo tính chính xác, độ tin cậy của dữ liệu nằm bên trong bảng. Nếu có bất kỳ vi phạm nào giữa ràng buộc và hành động dữ liệu thì hành động dữ liệu đó sẽ bị hủy bỏ.

Các ràng buộc có thể được chỉ định khi một bảng được tạo ra với câu lệnh CREATE TABLE hoặc sử dụng câu lệnh ALTER TABLE để tạo các ràng buộc ngay cả sau khi bảng được tạo.

## Cú pháp tạo ràng buộc

```
CREATE TABLE table_name(  
    column1 data_type constraint_type,  
    column2 data_type constraint_type,  
    ...  
    columnN data_type constraint_type  
) ;
```



Bất kỳ ràng buộc nào mà chúng đã định nghĩa có thể bị xóa bỏ bằng cách sử dụng lệnh ALTER TABLE với tùy chọn DROP CONSTRAINT.

## Cú pháp xóa ràng buộc

```
ALTER TABLE table_name DROP CONSTRAINT constraint_name;
```



Dưới đây là một số ràng buộc thường được sử dụng:

Loại ràng buộc	Mô tả
NOT NULL	Đảm bảo rằng một cột không thể có giá trị NULL
UNIQUE	Đảm bảo rằng tất cả các giá trị trong một cột không được phép trùng nhau.
PRIMARY KEY	Xác định mỗi dòng là duy nhất trong một bảng cơ sở dữ liệu.
FOREIGN KEY	Xây dựng một mối liên kết giữa hai bảng nhằm đảm bảo tính toàn vẹn dữ liệu.
CHECK	Giới hạn phạm vi giá trị được nhập vào bên trong cột.
DEFAULT	Cung cấp một giá trị mặc định cho một cột khi không có gì được chỉ định.
INDEX	Lập chỉ mục cho các cột để cải thiện tốc độ truy xuất dữ liệu từ cơ sở dữ liệu.
AUTO_INCREMENT	Tự động tăng giá trị khi bản ghi được thêm vào.

## 6.1 Ràng buộc NOT NULL

Khi chèn một hàng mới vào bên trong bảng, nếu chúng ta không xác định đầy đủ giá trị cho tất cả các cột thì mặc định những cột bị bỏ trống sẽ nhận giá trị NULL. Giá trị NULL có ý nghĩa là **không có dữ liệu gì cả**, nó hoàn toàn khác so với một chuỗi rỗng hoặc giá trị 0.

Ràng buộc NOT NULL là một kiểu ràng buộc, mà yêu cầu một trường hoặc một tập hợp các trường trong bảng không được lưu giá trị NULL. Điều này có nghĩa là bạn phải cung cấp một giá trị cho trường này khi thêm một hàng mới vào bảng, và không thể sửa đổi giá trị của trường này thành NULL trong hàng đã tồn tại.

Ràng buộc NOT NULL được sử dụng để bảo đảm tính nhất quán và hợp lý của dữ liệu trong cơ sở dữ liệu. Nó cũng giúp giảm khả năng xảy ra lỗi trong các truy vấn và các thao tác trên dữ liệu, vì nó yêu cầu tất cả các hàng trong bảng có giá trị cho trường này.

Ví dụ, đoạn mã bên dưới dùng để tạo một bảng SINHVIEN. Trong đó, hai cột MSSV và TEN bị thiết lập ràng buộc NOT NULL, đồng nghĩa với việc hai cột này không thể chứa giá trị NULL.

```
CREATE TABLE SINHVIEN (
    MSSV INT          NOT NULL,
    TEN VARCHAR (20)  NOT NULL,
    TUOI INT,
    DIACHI VARCHAR (30)
);
```

Có thể thêm và xóa ràng buộc NOT NULL sau khi tạo bảng bằng cách dùng lệnh ALTER TABLE.

### Cú pháp thêm ràng buộc NOT NULL

```
ALTER TABLE table_name MODIFY COLUMN column_name data_type NOT NULL;
```



Ví dụ, câu lệnh sau dùng để bổ sung ràng buộc NOT NULL cho cột DIACHI trong bảng SINHVIEN.

```
ALTER TABLE SINHVIEN MODIFY COLUMN DIACHI VARCHAR(30) NOT NULL;
```

### Cú pháp xóa ràng buộc NOT NULL

```
ALTER TABLE table_name MODIFY COLUMN column_name data_type;
```



Ví dụ, câu lệnh sau dùng để xóa ràng buộc NOT NULL trên cột DIACHI.

```
ALTER TABLE SINHVIEN MODIFY COLUMN DIACHI VARCHAR(30);
```

## 6.2 Ràng buộc UNIQUE

Ràng buộc UNIQUE yêu cầu mỗi giá trị trong cột hoặc tập hợp các cột được ràng buộc là duy nhất trong bảng. Tuy nhiên, cột được ràng buộc UNIQUE có thể có giá trị NULL.

- Nếu chúng ta thiết lập ràng buộc UNIQUE cho một cột thì dữ liệu trên các hàng của cột đó tuyệt đối không được trùng nhau.
- Nếu chúng ta thiết lập ràng buộc UNIQUE cho một tập hợp các cột thì dữ liệu trên các hàng của một tập hợp các cột đó tuyệt đối không được trùng nhau.

Để thiết lập ràng buộc UNIQUE cho một cột nào đó thì trong quá trình tạo bảng, chúng ta chỉ cần thêm từ khóa UNIQUE nằm phía sau tên cột.

Ví dụ, đoạn mã bên dưới dùng để tạo một bảng MONTHI. Trong đó, cột TENMT bị thiết lập ràng buộc UNIQUE, điều đó đồng nghĩa với việc tên của môn thi không được trùng nhau.

```
CREATE TABLE MONTHI (
    MAMT VARCHAR(4),
    TENMT VARCHAR(50) UNIQUE
);
```

### Cú pháp thiết lập ràng buộc UNIQUE

[CONSTRAINT constraint\_name] UNIQUE(column1, column2, column3,...)



Trong đó, cụm từ [CONSTRAINT constraint\_name] dùng để đặt tên cho ràng buộc, nếu chúng ta loại bỏ nó thì tên của cột column1 sẽ mặc định được sử dụng làm tên ràng buộc.

Cú pháp thiết lập ràng buộc UNIQUE cho một tập hợp các cột cũng có thể được dùng cho trường hợp thiết lập ràng buộc UNIQUE cho riêng một cột.

Ví dụ, trong bảng GIAOVIEN bên dưới, ta tạo một ràng buộc UNIQUE có tên là 'unique\_gv' dựa trên hai cột HOTEN và MABM. Điều đó đồng nghĩa với việc bên trong bảng GIAOVIEN không được phép tồn tại những giáo viên có cùng tên và cùng mã bộ môn.

```
CREATE TABLE GIAOVIEN (
    MAGV CHAR (6),
    HOTEN VARCHAR (40),
    TUOI INT,
    DIACHI VARCHAR(40),
    GIOITINH VARCHAR (3),
    MABM VARCHAR(4),
    CONSTRAINT unique_gv UNIQUE(HOTEN,MABM)
);
```

Lưu ý: Việc tạo ràng buộc UNIQUE cho một tập hợp các cột và tạo ràng buộc UNIQUE cho nhiều cột là hoàn toàn khác nhau. Ví dụ:

```
CREATE TABLE GIAOVIEN (
    MAGV CHAR (6),
    HOTEN VARCHAR (40),
    TUOI INT,
    MABM VARCHAR(4),
    UNIQUE(HOTEN,MABM)
);
```

```
CREATE TABLE GIAOVIEN (
    MAGV CHAR (6),
    HOTEN VARCHAR (40) UNIQUE,
    TUOI INT,
    MABM VARCHAR(4) UNIQUE
);
```

Hai đoạn mã nằm ở phía trên dùng để tạo bảng GIAOVIEN, chúng đều có ràng buộc UNIQUE liên quan đến hai cột HOTEN và MABM. Tuy nhiên, đối với đoạn mã thứ nhất thì bên trong bảng GIAOVIEN có thể có nhiều giáo viên trùng tên, có thể có nhiều giáo viên trùng mã bộ môn, chỉ riêng trường hợp không được phép tồn tại các giáo viên trùng tên lẫn mã bộ môn. Còn đối với đoạn mã thứ hai thì nó sẽ mang ý nghĩa là bên trong cái bảng GIAOVIEN không được phép có các giáo viên trùng tên, không được phép có các giáo viên trùng mã bộ môn.

Có thể thêm và xóa ràng buộc NOT NULL sau khi tạo bảng bằng cách dùng lệnh ALTER TABLE.

#### Cú pháp thêm ràng buộc UNIQUE

```
ALTER TABLE table_name ADD [CONSTRAINT constraint_name] UNIQUE(column1,...);
```

Ví dụ, để bổ sung ràng buộc UNIQUE cho cột MAGV chúng ta sử dụng câu lệnh sau:

```
ALTER TABLE GIAOVIEN ADD CONSTRAINT unique_MAGV UNIQUE(MAGV);
```

#### Cú pháp xóa ràng buộc UNIQUE

```
ALTER TABLE table_name DROP CONSTRAINT constraint_name;
```



Ví dụ, để xóa ràng buộc UNIQUE trên cột MAGV thì chúng ta sử dụng câu lệnh sau:

```
ALTER TABLE GIAOVIEN DROP INDEX unique_MAGV;
```

## 6.3 Ràng buộc PRIMARY KEY

Ràng buộc PRIMARY KEY (khóa chính) có chức năng tương tự như ràng buộc UNIQUE, nó được sử dụng để thiết lập việc dữ liệu trên các hàng không được phép trùng nhau và không được phép NULL.

Một ràng buộc PRIMARY KEY có thể được thiết lập cho một cột hoặc một tập hợp các cột.

- Nếu chúng ta thiết lập ràng buộc PRIMARY KEY cho một cột thì dữ liệu trên các hàng của cột đó tuyệt đối không được trùng nhau.
- Nếu chúng ta thiết lập ràng buộc PRIMARY KEY cho một tập hợp các cột thì dữ liệu trên các hàng của một tập hợp các cột đó tuyệt đối không được trùng nhau.

Trên một bảng chỉ được phép tồn tại tối đa một ràng buộc PRIMARY KEY.

Thông thường thì ràng buộc PRIMARY KEY chỉ dùng để thiết lập cho riêng một cột (điển hình như các cột liên quan đến ID, mã số,...) để nhận diện một hàng.

Để thiết lập ràng buộc PRIMARY KEY cho một cột nào đó thì trong quá trình tạo bảng, chúng ta chỉ cần thêm từ khóa PRIMARY KEY nằm phía sau tên cột.

Ví dụ, đoạn mã bên dưới dùng để tạo một bảng MONTHI. Trong đó, cột MAMT bị thiết lập ràng buộc PRIMARY KEY, điều đó đồng nghĩa với việc mã môn thi của các môn không được phép trùng nhau.

```
CREATE TABLE MONTHI (
    MAMT VARCHAR(4) PRIMARY KEY,
    TENMT VARCHAR(50)
);
```

Cú pháp sau dùng để thiết lập ràng buộc PRIMARY KEY cho một tập hợp các cột.

### Cú pháp thiết lập ràng buộc PRIMARY KEY

```
[CONSTRAINT constraint_name] PRIMARY KEY(column1, column2, column3, ...)
```



Ví dụ, trong bảng GIAOVIEN bên dưới, ta tạo một ràng buộc PRIMARY KEY tên là 'pk\_gv' dựa trên hai cột HOTEN và MABM. Điều đó đồng nghĩa với việc bên trong bảng GIAOVIEN không được phép tồn tại những giáo viên có cùng tên và cùng mã bộ môn.

```
CREATE TABLE GIAOVIEN (
    MAGV CHAR (6),
    HOTEN VARCHAR (40),
    TUOI INT,
    DIACHI VARCHAR(40),
    GIOITINH VARCHAR (3),
    MABM VARCHAR(4),
    CONSTRAINT pk_gv PRIMARY KEY(HOTEN,MABM)
);
```

Có thể thêm và xóa ràng buộc NOT NULL sau khi tạo bảng bằng cách dùng lệnh ALTER TABLE.

#### Cú pháp thêm ràng buộc PRIMARY KEY

```
ALTER TABLE table_name ADD [CONSTRAINT constraint_name] PRIMARY KEY(column1,...);
```

Ví dụ, để thiết lập ràng buộc PRIMARY KEY là MAGV chúng ta sử dụng câu lệnh sau:

```
ALTER TABLE GIAOVIEN ADD CONSTRAINT pk_gv PRIMARY KEY(MAGV);
```

#### Cú pháp xóa ràng buộc PRIMARY KEY

```
ALTER TABLE table_name DROP CONSTRAINT constraint_name;
```

hoặc

```
ALTER TABLE table_name DROP PRIMARY KEY;
```



Ví dụ, để xóa ràng buộc PRIMARY KEY ra khỏi bảng GIAOVIEN thì chúng ta sử dụng lệnh sau:

```
ALTER TABLE GIAOVIEN DROP CONSTRAINT pk_gv;
```

hoặc lệnh

```
ALTER TABLE GIAOVIEN DROP PRIMARY KEY;
```

## 6.4 Ràng buộc FOREIGN KEY

Ràng buộc FOREIGN KEY (khóa ngoại) được dùng để xây dựng một mối liên kết giữa hai cái bảng nhằm đảm bảo tính "tồn vẹn dữ liệu" của các bảng (khi trong hai bảng có một cái bảng tham chiếu đến dữ liệu của bảng còn lại).

Trong hai bảng, bảng chứa ràng buộc FOREIGN KEY được gọi là *bảng con*, còn bảng chứa khóa ứng viên thì được gọi là *bảng tham chiếu* hoặc *bảng cha*.

### Cú pháp thiết lập ràng buộc FOREIGN KEY

```
[CONSTRAINT constraint_name] FOREIGN KEY(column1) REFERENCES table_name(column2)
```

Trong đó

- column1 là tên của cột (bên trong bảng con) mà chúng ta muốn thiết lập ràng buộc FOREIGN KEY.
- table\_name là tên của bảng tham chiếu đến.
- column2 là tên của cột (bên trong bảng tham chiếu) mà cột column1 tham chiếu đến, ngoài ra column2 còn phải là khóa chính của bảng tham chiếu.



Ví dụ,

```
CREATE TABLE BOMON(
    MABM VARCHAR(6) PRIMARY KEY,
    TENBM VARCHAR(20)
);

CREATE TABLE GIAOVIEN(
    MAGV CHAR (6) PRIMARY KEY,
    HOTEN VARCHAR (40),
    TUOI INT NOT NULL,
    DIACHI VARCHAR(40),
    GIOITINH VARCHAR (3),
    MABM VARCHAR(4),
    CONSTRAINT fk_mabm FOREIGN KEY (MABM) REFERENCES BOMON(MABM)
);
```

Cột MABM trong bảng GIAOVIEN tham chiếu dữ liệu từ cột MABM trong bảng BOMON, cho biết những giáo viên nào thuộc bộ môn nào (nếu thêm vào bảng GIAOVIEN những giáo viên có mã bộ môn không tồn tại trong bảng BOMON thì những giáo viên đó sẽ không hợp lệ).

Để tránh việc bảng GIAOVIEN chứa những giáo viên không hợp lệ thì chúng ta cần phải thiết lập ràng buộc FOREIGN KEY cho cột MABM, ràng buộc này đảm bảo việc ngăn chặn những hành động không hợp lệ, điển hình như:

- Không cho phép thêm những giáo viên có MABM không tồn tại trong bảng BOMON.
- Không cho phép xóa những bộ môn mà MABM có tồn tại trong bảng GIAOVIEN.
- Không cho phép xóa bảng BOMON khi bảng GIAOVIEN vẫn còn đang tồn tại.
- ...

Đối với trường hợp bảng đã được tạo trước đó, nhưng do quên thiết lập ràng buộc FOREIGN KEY cho cột nên bây giờ chúng ta muốn bổ sung, để làm được điều đó thì chúng ta sử dụng lệnh ALTER TABLE.

### Cú pháp thêm ràng buộc FOREIGN KEY

```
ALTER TABLE table1 ADD [CONSTRAINT constraint_name]  
FOREIGN KEY(column1) REFERENCES table2(column2);
```



Ví dụ, câu lệnh bên dưới dùng để thêm ràng buộc FOREIGN KEY cho cột MABM (trong bảng GIAOVIEN) sau khi đã tạo bảng này

```
ALTER TABLE GIAOVIEN ADD CONSTRAINT fk_mabm  
FOREIGN KEY (MABM) REFERENCES BOMON(MABM);
```

Chúng ta có thể xóa ràng buộc FOREIGN KEY bằng cách sử dụng lệnh ALTER TABLE.

### Cú pháp xóa ràng buộc FOREIGN KEY

```
ALTER TABLE table_name DROP CONSTRAINT constraint_name;  
hoặc  
ALTER TABLE table_name DROP FOREIGN KEY constraint_name;
```



Ví dụ, để xóa ràng buộc FOREIGN KEY trên cột MABM ra khỏi bảng GIAOVIEN thì chúng ta sử dụng câu lệnh sau:

```
ALTER TABLE GIAOVIEN DROP CONSTRAINT fk_mabm;
```

hoặc lệnh

```
ALTER TABLE GIAOVIEN DROP FOREIGN KEY fk_mabm;
```

## 6.5 Ràng buộc CHECK

Ràng buộc CHECK được sử dụng để thiết lập việc giới hạn phạm vi giá trị được nhập vào bên trong cột. Để thiết lập một ràng buộc CHECK thì chúng ta sử dụng cú pháp như sau:

### Cú pháp thiết lập ràng buộc CHECK

```
[CONSTRAINT constraint_name] CHECK (condition)
```



Ví dụ, đoạn mã bên dưới dùng để tạo một bảng SINHVIEN. Trong đó, chúng ta thiết lập ràng buộc CHECK cho cột GIOITINH với tên là 'check\_gt', ràng buộc này bắt buộc các sinh viên được thêm vào bên trong bảng phải có giới tính là nam hoặc là nữ.

```
CREATE TABLE SINHVIEN (
    MSSV INT,
    TEN VARCHAR (20),
    TUOI INT,
    GIOITINH VARCHAR (3),
    CONSTRAINT check_gt CHECK (GIOITINH IN ('Nam', 'Nu'))
);
```

Đối với trường hợp bảng đã được tạo trước đó, nhưng do quên thiết lập ràng buộc CHECK cho cột nên bây giờ chúng ta muốn bổ sung, để làm được điều đó thì chúng ta sử dụng lệnh ALTER TABLE.

### Cú pháp thêm ràng buộc CHECK

```
ALTER TABLE table_name ADD [CONSTRAINT constraint_name] CHECK (condition);
```



Ví dụ, câu lệnh bên dưới dùng để thêm ràng buộc CHECK cho cột GIOITINH sau khi đã tạo bảng này

```
ALTER TABLE SINHVIEN ADD CONSTRAINT check_gt CHECK (GIOITINH IN ('Nam', 'Nu'));
```

Chúng ta có thể xóa ràng buộc CHECK bằng cách sử dụng lệnh ALTER TABLE.

### Cú pháp xóa ràng buộc CHECK

```
ALTER TABLE table_name DROP CONSTRAINT constraint_name;
```



Ví dụ, để xóa ràng buộc CHECK trên cột GIOITINH ra khỏi bảng SINHVIEN, chúng ta sử dụng lệnh sau:

```
ALTER TABLE SINHVIEN DROP CONSTRAINT check_gt;
```

## 6.6 Ràng buộc DEFAULT

Ràng buộc DEFAULT được dùng để thiết lập một giá trị mặc định cho cột, điều đó có nghĩa là khi chúng ta chèn một hàng mới vào bên trong bảng mà không gán giá trị cho cột này thì cột này sẽ nhận giá trị mặc định.

Ví dụ, đoạn mã bên dưới dùng để tạo một bảng SINHVIEN. Trong đó, chúng ta thiết lập ràng buộc DEFAULT cho cột TUOI, nếu chúng ta chèn một sinh viên mà không gán giá trị cho cột TUOI thì mặc định sinh viên sẽ có tuổi là 22.

```
CREATE TABLE SINHVIEN (
    MSSV INT,
    TEN VARCHAR (20),
    TUOI INT DEFAULT 22,
    GIOITINH VARCHAR (3)
);
```

Đối với trường hợp bảng đã được tạo trước đó, nhưng do quên thiết lập ràng buộc DEFAULT cho cột nên bây giờ chúng ta muốn bổ sung, để làm được điều đó thì chúng ta sử dụng lệnh ALTER TABLE.

### Cú pháp thêm ràng buộc DEFAULT

MySQL: ALTER TABLE table_name ALTER column_name SET DEFAULT value;	♥
SQL Server: ALTER TABLE t_name ADD CONSTRAINT c_name DEFAULT value FOR col_name;	
Oracle: ALTER TABLE table_name MODIFY column_name DEFAULT value;	♥

Ví dụ, câu lệnh sau dùng để thiết lập giá trị mặc định cho cột TUOI là 22 (trong bảng SINHVIEN).

```
ALTER TABLE SINHVIEN ALTER TUOI SET DEFAULT 22;
```

Chúng ta có thể xóa ràng buộc DEFAULT bằng cách sử dụng lệnh ALTER TABLE.

### Cú pháp xóa ràng buộc DEFAULT

MySQL: ALTER TABLE table_name ALTER column_name DROP DEFAULT;	♥
SQL Server: DROP DEFAULT c_name;	
Oracle: ALTER TABLE table_name MODIFY column_name DEFAULT NULL;	♥

Ví dụ, để xóa ràng buộc DEFAULT trên cột TUOI ra khỏi bảng SINHVIEN trong MySQL, chúng ta sử dụng câu truy vấn sau đây:

```
ALTER TABLE SINHVIEN ALTER TUOI DROP DEFAULT;
```

## 6.7 Câu lệnh INDEX

Lệnh INDEX được sử dụng để tạo ra một chỉ mục cho một cột hoặc nhiều cột trong một bảng. Chỉ mục được sử dụng để cải thiện hiệu suất trong việc truy vấn dữ liệu và làm cho nó nhanh hơn để tìm kiếm dữ liệu trong bảng.

Việc lập chỉ mục trên các cột sẽ giúp cải thiện tốc độ truy vấn dữ liệu từ cơ sở dữ liệu. Tuy nhiên, khi cập nhật bảng có chỉ mục sẽ chậm hơn so với bảng không có chỉ mục, vì chỉ mục cũng cần phải cập nhật, do đó chỉ nên lập chỉ mục trên những cột được tìm kiếm thường xuyên.

Để lập chỉ mục cho cột thì chúng ta sử dụng lệnh CREATE INDEX với cú pháp như sau:

### Cú pháp lập chỉ mục

```
CREATE INDEX index_name ON table_name (column1, column2, column3,...);
```



Ví dụ, để thiết lập chỉ mục (có tên là 'index\_sv') trên hai cột TEN và DIACHI của bảng SINHVIEN ta sử dụng câu lệnh sau:

```
CREATE INDEX index_sv ON SINHVIEN (TEN, DIACHI);
```

Chúng ta có thể xóa một chỉ mục ra khỏi bảng bằng cách sử dụng lệnh ALTER TABLE.

### Cú pháp xóa chỉ mục

```
ALTER TABLE table_name DROP INDEX index_name;
```



Ví dụ, để xóa có tên là 'index\_sv' ra khỏi bảng SINHVIEN, chúng ta sử dụng câu lệnh sau:

```
ALTER TABLE SINHVIEN DROP INDEX index_sv;
```

## 6.8 AUTO INCREMENT

Thuộc tính AUTO\_INCREMENT thường được dùng trên những cột là khóa chính và có *kiểu dữ liệu là số nguyên*, chức năng của nó là tự động gán giá trị cho cột khi chúng ta thêm hàng mới vào bên trong bảng (giá trị của hàng đầu tiên được thêm vào bảng là 1, giá trị này sẽ tự động tăng lên một sau mỗi hàng được thêm vào bảng).

Ví dụ, đoạn mã bên dưới được dùng để tạo một bảng DETAI. Trong đó, cột stt được thiết lập thuộc tính AUTO\_INCREMENT, khi chúng ta chèn một hàng mới vào bên trong bảng thì cột này sẽ tự động được gán giá trị (giá trị sẽ tự động tăng thêm một sau mỗi hàng được chèn vào bảng).

```
CREATE TABLE DETAI(
    STT INT PRIMARY KEY AUTO_INCREMENT,
    TENDT VARCHAR(255),
    THOIGIAN DATE
);
```

Sau đó ta thực hiện các câu lệnh INSERT sau để thêm dữ liệu vào bảng DETAI

```
INSERT INTO DETAI VALUES
('Tim hieu ngon ngu SQL', '2022-12-01'),
('Cong nghe nhan dang khuon mat', '2023-01-10'),
('What is Machine learning?', '2023-05-01');
```

Kết quả sẽ trả về bảng DETAI như bên dưới:

STT	TENDT	THOIGIAN
1	Tim hieu ngon ngu SQL	2022-12-01
2	Cong nghe nhan dang khuon mat	2023-01-10
3	What is Machine learning?	2023-05-01

Nếu muốn thiết lập giá trị AUTO\_INCREMENT cho hàng chuẩn bị được thêm vào bảng thì chúng ta có thể sử dụng cú pháp sau:

### Khởi tạo giá trị AUTO\_INCREMENT

```
ALTER TABLE table_name AUTO_INCREMENT = value;
```



Ví dụ, chúng ta thực hiện các câu lệnh sau để tiếp tục thêm dữ liệu vào bảng DETAI

```
ALTER TABLE DETAI AUTO_INCREMENT = 50;
INSERT INTO DETAI VALUES ('Dien toan dam may', '2023-08-06'),
('Xu ly anh voi Python', '2023-09-10'),
('Tri tue nhan tao la gi?', '2023-09-11');
```

Kết quả sẽ trả về bảng DETAI như bên dưới

STT	TENDT	THOIGIAN
1	Tim hieu ngon ngu SQL	2022-12-01
2	Cong nghe nhan dang khuon mat	2023-01-10
3	What is Machine learning?	2023-05-01
50	Dien toan dam may	2023-08-06
51	Xu ly anh voi Python	2023-09-10
52	Tri tue nhan tao la gi?	2023-09-21

Chúng ta có thể thêm hoặc xóa một chỉ mục bằng cách sử dụng câu lệnh ALTER TABLE.

#### Thêm thuộc tính AUTO\_INCREMENT

```
ALTER TABLE table_name MODIFY COLUMN column_name data_type AUTO_INCREMENT; 
```

Ví dụ: Để thêm thuộc tính AUTO\_INCREMENT cho cột STT thì ta sử dụng câu lệnh sau:

```
ALTER TABLE DETAI MODIFY COLUMN STT INT AUTO_INCREMENT;
```

#### Xóa thuộc tính AUTO\_INCREMENT

```
ALTER TABLE table_name MODIFY COLUMN column_name data_type; 
```

Ví dụ: Để xóa thuộc tính AUTO\_INCREMENT nằm ở trên cột STT thì ta sử dụng câu lệnh sau:

```
ALTER TABLE DETAI MODIFY COLUMN STT INT;
```

# CHƯƠNG 7. CÁC HÀM CƠ BẢN TRONG MYSQL

Trong hệ quản trị cơ sở dữ liệu MySQL xây dựng sẵn khá nhiều hàm (function) giúp chúng ta giải quyết những công việc mà chúng ta phải thường xuyên thực hiện. Dưới đây là bảng mô tả sơ qua về chức năng của một số hàm cơ bản thường sử dụng.

## Các hàm làm việc với số

Tên hàm	Mô tả
SUM	Tính tổng của các giá trị số bên trong cột.
MAX	Trả về giá trị lớn nhất trong cột.
MIN	Trả về giá trị nhỏ nhất trong cột.
AVG	Tính giá trị trung bình của các số bên trong cột.
COUNT	Đếm số lượng hàng (bản ghi) được trả về từ lệnh truy vấn dữ liệu.
ABS	Trả về giá trị tuyệt đối của một số.
CEIL	Trả về giá trị nguyên nhỏ nhất lớn hơn hoặc bằng một số.
FLOOR	Trả về giá trị nguyên lớn nhất bằng hoặc nhỏ hơn một số.
RAND	Trả về một số ngẫu nhiên trong khoảng từ 0 đến 1.
ROUND	Trả về một số được làm tròn đến một số vị trí thập phân nhất định.

## Các hàm làm việc với chuỗi

Tên hàm	Mô tả
CHAR_LENGTH	Trả về độ dài (số lượng ký tự) của một chuỗi.
CONCAT	Nối hai hoặc nhiều chuỗi lại với nhau.
CONCAT_WS	Nối hai hoặc nhiều chuỗi lại với nhau (có kèm theo 'phân cách' giữa các chuỗi).
FORMAT	Định dạng một số thành dạng #,###,###,###.##
LOWER	Chuyển đổi các ký tự bên trong chuỗi sang dạng chữ thường.
UPPER	Chuyển đổi các ký tự bên trong chuỗi sang dạng chữ in hoa.
REPLACE	Thay tất cả các lần xuất hiện của một chuỗi được chỉ định bằng một chuỗi mới.
REVERSE	Đảo ngược thứ tự của các ký tự bên trong chuỗi.
SUBSTRING	Trích xuất một chuỗi con từ một chuỗi.
TRIM	Loại bỏ tất cả các ký tự được chỉ định từ đầu hoặc cuối chuỗi.
STRCMP	Kiểm tra xem hai chuỗi có giống nhau hay không.
SPACE	Trả về một chuỗi có số lượng khoảng trắng được chỉ định.
LEFT	Trích xuất một chuỗi con từ một chuỗi, bắt đầu từ ký tự ngoài cùng bên trái.
RIGHT	Trích xuất một chuỗi con từ một chuỗi, bắt đầu từ ký tự ngoài cùng bên phải.

---

## Các hàm liên quan đến giá trị rỗng

Tên hàm	Mô tả
COALESCE	Trả về giá trị khác NULL đầu tiên trong danh sách.
IFNULL	Kiểm tra một giá trị, nếu giá trị là NULL thì hàm sẽ trả về một giá trị chỉ định, còn nếu giá trị không phải là NULL thì hàm sẽ trả về chính giá trị đó.
ISNULL	Kiểm tra một giá trị, nếu giá trị là NULL thì hàm sẽ trả về 1, còn nếu giá trị không phải là NULL thì hàm sẽ trả về 0.
NULIF	Kiểm tra hai giá trị, nếu hai giá trị giống nhau thì hàm sẽ trả về NULL, nếu hai giá trị khác nhau thì hàm sẽ trả về giá trị thứ nhất.

## Một số hàm quan trọng khác

Tên hàm	Mô tả
CURDATE	Trả về thời gian hiện tại
IF	Kiểm tra một biểu thức điều kiện, nếu đúng thì trả về giá trị được chỉ định thứ nhất, nếu sai thì trả về giá trị được chỉ định thứ hai.
LAST_INSERT_ID	Trả về giá trị id AUTO_INCREMENT của hàng cuối cùng.

## 7.1 Hàm SUM

Hàm SUM dùng để tính tổng của các giá trị số bên trong cột.

Ví dụ: Cho bảng dữ liệu SINHVIEN sau đây:

MSSV	TEN	TUOI	DIACHI	GIOITINH
12150	Doan Du	28	HCM	Nam
12151	Vuong Ngu Yen	25	Long An	Nu
12152	Nham Doanh Doanh	26	Tay Ninh	Nu
12153	Ly Mac Sau	23	Binh Duong	Nu
12154	Au Duong Phong	23	HCM	Nam

Nếu muốn tính tổng của tất cả tuổi trong cột TUOI, ta sử dụng câu lệnh sau:

```
SELECT SUM(TUOI) AS TONGTUOI FROM SINHVIEN; -- Kết quả: 125
```

Ví dụ: Cho bảng dữ liệu KETQUA sau đây:

MSSV	MAMT	DIEM
12150	CSDL	8.5
12150	THDC	9.0
12152	CSDL	9.0
12154	CSDL	9.5

Nếu muốn tính tổng điểm theo từng sinh viên, ta sử dụng GROUP BY, câu lệnh như sau:

```
SELECT MSSV, SUM(DIEM)
FROM KETQUA GROUP BY MSSV;
```

Kết quả:

MSSV	DIEM
12150	17.5
12152	9.0
12154	9.5

## 7.2 Hàm MAX

Hàm MAX dùng để lấy giá trị lớn nhất bên trong cột.

Ví dụ: Cho bảng dữ liệu SINHVIEN sau đây:

MSSV	TEN	TUOI	DIACHI	GIOITINH
12150	Doan Du	28	HCM	Nam
12151	Vuong Ngu Yen	25	Long An	Nu
12152	Nham Doanh Doanh	26	Tay Ninh	Nu
12153	Ly Mac Sau	23	Binh Duong	Nu
12154	Au Duong Phong	23	HCM	Nam

Nếu muốn lấy giá trị lớn nhất trong cột TUOI, ta sử dụng câu lệnh sau:

```
SELECT MAX(TUOI) AS TUOI_LON_NHAT FROM SINHVIEN; -- Kết quả: 28
```

Ví dụ: Cho bảng dữ liệu KETQUA sau đây:

MSSV	MAMT	DIEM
12150	CSDL	8.5
12150	THDC	9.0
12152	CSDL	9.0
12154	CSDL	9.5

Nếu muốn tìm điểm lớn nhất theo từng sinh viên, ta sử dụng GROUP BY, câu lệnh như sau:

```
SELECT MSSV, MAX(DIEM) AS DIEMMAX
FROM KETQUA GROUP BY MSSV;
```

Kết quả:

MSSV	DIEMMAX
12150	9.0
12152	9.0
12154	9.5

## 7.3 Hàm MIN

Hàm MIN dùng để lấy giá trị nhỏ nhất bên trong cột.

Ví dụ: Cho bảng dữ liệu SINHVIEN sau đây:

MSSV	TEN	TUOI	DIACHI	GIOITINH
12150	Doan Du	28	HCM	Nam
12151	Vuong Ngu Yen	25	Long An	Nu
12152	Nham Doanh Doanh	26	Tay Ninh	Nu
12153	Ly Mac Sau	23	Binh Duong	Nu
12154	Au Duong Phong	23	HCM	Nam

Nếu muốn lấy giá trị nhỏ nhất trong cột TUOI, ta sử dụng câu lệnh sau:

```
SELECT MIN(TUOI) AS TUOI_NHO_NHAT FROM SINHVIEN; -- Kết quả: 23
```

Ví dụ: Cho bảng dữ liệu KETQUA sau đây:

MSSV	MAMT	DIEM
12150	CSDL	8.5
12150	THDC	9.0
12152	CSDL	9.0
12154	CSDL	9.5

Nếu muốn tìm điểm nhỏ nhất theo từng sinh viên, ta sử dụng GROUP BY, câu lệnh như sau:

```
SELECT MSSV, MIN(DIEM) AS DIEMMIN
FROM KETQUA GROUP BY MSSV;
```

Kết quả:

MSSV	DIEMMIN
12150	8.5
12152	9.0
12154	9.5

## 7.4 Hàm AVG

Hàm AVG dùng để tìm giá trị trung bình của một cột.

Ví dụ: Cho bảng dữ liệu SINHVIEN sau đây:

MSSV	TEN	TUOI	DIACHI	GIOITINH
12150	Doan Du	28	HCM	Nam
12151	Vuong Ngu Yen	25	Long An	Nu
12152	Nham Doanh Doanh	26	Tay Ninh	Nu
12153	Ly Mac Sau	23	Binh Duong	Nu
12154	Au Duong Phong	23	HCM	Nam

Nếu muốn lấy giá trị tuổi trung bình, ta sử dụng câu lệnh sau:

```
SELECT AVG(TUOI) AS TUOI_TB FROM SINHVIEN; -- Kết quả: 25
```

Ví dụ: Cho bảng dữ liệu KETQUA sau đây:

MSSV	MAMT	DIEM
12150	CSDL	8.5
12150	THDC	9.0
12152	CSDL	9.0
12154	CSDL	9.5

Nếu muốn tìm điểm trung bình theo từng sinh viên, ta sử dụng GROUP BY, câu lệnh như sau:

```
SELECT MSSV, AVG(DIEM) AS DIEMTB
FROM KETQUA GROUP BY MSSV;
```

Kết quả:

MSSV	DIEMTB
12150	8.75
12152	9.0
12154	9.5

## 7.5 Hàm COUNT

Hàm COUNT dùng để đếm số lượng hàng (bản ghi) được trả về từ câu lệnh SELECT.

Ví dụ: Cho bảng dữ liệu SINHVIEN sau đây:

MSSV	TEN	TUOI	DIACHI	GIOITINH
12150	Doan Du	28	HCM	Nam
12151	Vuong Ngu Yen	25	Long An	Nu
12152	Nham Doanh Doanh	26	Tay Ninh	Nu
12153	Ly Mac Sau	23	Binh Duong	Nu
12154	Au Duong Phong	23	HCM	Nam

Nếu muốn đếm số lượng sinh viên có trong danh sách, ta sử dụng câu lệnh sau:

```
SELECT COUNT(*) AS SL FROM SINHVIEN; -- Kết quả: 5
```

Ví dụ: Cho bảng dữ liệu KETQUA sau đây:

MSSV	MAMT	DIEM
12150	CSDL	8.5
12150	THDC	9.0
12152	CSDL	9.0
12154	CSDL	9.5

Nếu muốn đếm số lượng bài thi của từng sinh viên, ta sử dụng GROUP BY, câu lệnh như sau:

```
SELECT MSSV, COUNT(*) AS SL
FROM KETQUA GROUP BY MSSV;
```

Kết quả:

MSSV	SL
12150	2
12152	1
12154	1

## 7.6 Hàm LENGTH, CHAR LENGTH

Hàm CHAR\_LENGTH(str) trả về độ dài của 1 chuỗi (số ký tự trong chuỗi đó), kể cả khoảng trắng, hàm này còn được gọi là hàm đếm số ký tự.

Hàm LENGTH(str) trả về độ dài của 1 chuỗi theo byte.

Ví dụ:

```
SET @str:='Xin chào';
SELECT CHAR_LENGTH(@str); -- Kết quả: 8
SELECT LENGTH(@str); -- Kết quả: 9
```

Nếu chuỗi là NULL thì hàm CHAR\_LENGTH sẽ trả về kết quả là NULL. Nếu giá trị truyền vào không phải là chuỗi thì nó sẽ được đổi sang định dạng chuỗi.

Ví dụ:

```
SELECT CHAR_LENGTH(NULL); -- Kết quả: NULL
SELECT CHAR_LENGTH(''); -- Kết quả: 0
SELECT CHAR_LENGTH(000042) -- Kết quả: 2, khi đổi so sang chuoi là '42'
SELECT CHAR_LENGTH(4260) -- Kết quả: 4, khi đổi so sang chuoi là '4260'
SELECT CHAR_LENGTH('hello world! '); -- Kết quả: 15
```

Để sử dụng hàm CHAR\_LENGTH giống như hàm LEN trong SQL SERVER (hàm trả về ký tự trong chuỗi, không tính khoảng trắng ở bên phải) ta dùng cú pháp như sau:

```
SELECT CHAR_LENGTH(RTRIM('hello world! ')); -- Kết quả: 12
```

## 7.7 Hàm CONCAT, CONCAT WS

Hàm CONCAT(str1,str2,...) trả về chuỗi là kết quả của việc nối chuỗi các tham số.

Hàm CONCAT\_WS(separator,str1,str2,...) là viết tắt của Concatenate With Separator và là một dạng hàm CONCAT() đặc biệt. Tham số đầu tiên là Separator cho các tham số còn lại. Separator này được thêm vào giữa các chuỗi để được nối chuỗi. Separator có thể là một chuỗi. Nếu Separator là NULL thì kết quả trả về là NULL.

Ví dụ:

```
SELECT CONCAT('My', 'S', 'QL'); -- Kết quả: 'MySQL'
SELECT CONCAT_WS(',','Hello','World'); -- Kết quả: 'Hello,World'
SELECT CONCAT_WS(NULL,'Hello','World'); -- Kết quả: NULL
```

## 7.8 Hàm LOWER, UPPER

Hàm LOWER(str) trả về chuỗi str với tất cả ký tự đã được chuyển đổi thành chữ thường. Hàm UPPER(str) trả về chuỗi str với tất cả ký tự đã được chuyển đổi thành chữ hoa.

Ví dụ:

```
SELECT LOWER('Hello World'); -- Kết quả: 'hello world'
SELECT UPPER('Hello World'); -- Kết quả: 'HELLO WORLD'
```

## 7.9 Hàm FORMAT

Chức năng của hàm FORMAT(X,D) là dùng để định dạng một số thành dạng #,###,###,###.##, rất thích hợp trong việc hiển thị tiền tệ.

Định dạng số X được làm tròn về D vị trí sau dấu phẩy, và trả về kết quả dưới dạng một chuỗi. Nếu D là 0, thì kết quả không có dấu thập phân hay phần thập phân.

Ví dụ:

```
SELECT FORMAT(652648.8543, 2); -- Kết quả: '652,648.85'
SELECT FORMAT(652648.8543, 0); -- Kết quả: '652,649'
```

## 7.10 Hàm REPLACE

Chức năng của hàm REPLACE(string, old\_str, new\_str) là dùng để thay thế tất cả các lần xuất hiện của một chuỗi con (old\_str) bên trong chuỗi cha (string) bằng một chuỗi mới (new\_str).

Ví dụ:

```
SELECT REPLACE('Hello World', 'World', 'MySQL'); -- Kết quả: 'Hello MySQL'
```

## 7.11 Hàm COALESCE

Hàm COALESCE là dùng để trả về giá trị khác NULL đầu tiên trong một danh sách các giá trị.

Ví dụ:

```
SELECT COALESCE(NULL, NULL, 'MySQL', 'C++', NULL, 'JAVA'); -- Kết quả: 'MySQL'
```

## 7.12 Hàm SUBSTRING, SUBSTR

Hàm SUBSTRING dùng để trích xuất một chuỗi con từ một chuỗi.

Cú pháp:

```
SUBSTRING( string, start_position, [ length ] )
SUBSTRING( string FROM start_position [ FOR length ] )
```

Trong đó:

- **string:** Chuỗi nguồn.
- **start\_position:** Vị trí bắt đầu trích xuất. Vị trí đầu tiên trong chuỗi luôn là 1. Nếu **start\_position** là một số âm, thì hàm SUBSTRING bắt đầu từ cuối chuỗi và đếm ngược
- **length:** Không bắt buộc. Đây là số lượng ký tự để trích xuất. Nếu tham số này bị bỏ qua, hàm SUBSTRING sẽ trả về toàn bộ chuỗi.

Ví dụ:

```
SELECT SUBSTRING('Hello World', 4); -- Kết quả: 'lo World'
SELECT SUBSTRING('Hello World' FROM 4); -- Kết quả: 'lo World'
SELECT SUBSTRING('Hello World' , 1 , 4); -- Kết quả: 'Hell'
SELECT SUBSTRING('Hello World' FROM 1 FOR 4); -- Kết quả: 'Hell'
SELECT SUBSTRING('Hello World' , -4 , 4); -- Kết quả: 'orld'
SELECT SUBSTRING('Hello World' , 40); -- Kết quả: ''
```

Hàm SUBSTR được dùng để trích xuất một chuỗi con từ một chuỗi.

Cú pháp:

```
SUBSTR( string, start_position, [ length ] )
SUBSTR( string FROM start_position [ FOR length ] )
```

Ví dụ:

```
SELECT SUBSTR('Hello World', 4); -- Kết quả: 'lo World'
SELECT SUBSTR('Hello World' FROM 4); -- Kết quả: 'lo World'
SELECT SUBSTR('Hello World' , 1 , 4); -- Kết quả: 'Hell'
SELECT SUBSTR('Hello World' FROM 1 FOR 4); -- Kết quả: 'Hell'
SELECT SUBSTR('Hello World' , -4 , 4); -- Kết quả: 'orld'
SELECT SUBSTR('Hello World' , 40); -- Kết quả: ''
```

## 7.13 Hàm STRCMP

Hàm STRCMP( string1, string2 ) kiểm tra xem hai chuỗi có giống nhau hay không.

Trong đó:

- **string1 và string2:** Hai chuỗi được so sánh với nhau.
- Nếu **string1 và string2** giống nhau, hàm STRCMP sẽ trả về 0.
- Nếu **string1** nhỏ hơn **string2**, hàm STRCMP sẽ trả về -1.
- Nếu **string1** lớn hơn **string2**, hàm STRCMP sẽ trả về 1.

Ví dụ:

```
SELECT STRCMP('MySQL', 'MySQL'); -- Kết quả: 0
SELECT STRCMP('SQL', 'MySQL'); -- Kết quả: -1
SELECT STRCMP('MySQL', 'SQL'); -- Kết quả: 1
```

## 7.14 Hàm SPACE

Hàm SPACE(number) trả về một chuỗi có số lượng khoảng trắng được chỉ định.

```
SELECT SPACE(5); -- Kết quả: ' '
SELECT SPACE(10); -- Kết quả: ' '
```

## 7.15 Hàm LEFT, RIGHT

Hàm LEFT(string, number\_of\_characters) trích xuất một chuỗi con từ một chuỗi, bắt đầu từ ký tự ngoài cùng bên trái. Hàm RIGHT(string, number\_of\_characters) trích xuất một chuỗi con từ một chuỗi, bắt đầu từ ký tự ngoài cùng bên phải.

Nếu **number\_of\_characters** vượt quá độ dài **string** thì trả về **string** ban đầu.

```
SELECT LEFT('Hello MySQL', 1); -- Kết quả: 'H'
SELECT LEFT('Hello MySQL', 4); -- Kết quả: 'Hell'
SELECT LEFT('Hello MySQL', 40); -- Kết quả: 'Hello MySQL'

SELECT RIGHT('Hello MySQL', 1); -- Kết quả: 'L'
SELECT RIGHT('Hello MySQL', 4); -- Kết quả: 'ySQL'
SELECT RIGHT('Hello MySQL', 40); -- Kết quả: 'Hello MySQL'
```

## 7.16 Hàm TRIM, LTRIM, RTRIM

Hàm TRIM dùng để loại bỏ tất cả các ký tự được chỉ định từ đầu hoặc cuối chuỗi.

Cú pháp: TRIM( [ LEADING or TRAILING or BOTH ] [ trim\_character FROM ] string )

Trong đó:

- LEADING: Không bắt buộc. Loại bỏ trim\_character từ phía trước của chuỗi.
- TRAILING: Không bắt buộc. Loại bỏ trim\_character khỏi cuối chuỗi.
- BOTH: Không bắt buộc. Các ký tự sẽ được loại bỏ khỏi chuỗi. Nếu tham số này bị bỏ qua, nó sẽ xóa các ký tự khoảng trắng khỏi chuỗi.
- string: Các chuỗi để cắt.

Lưu ý:

- Nếu bạn không chỉ định giá trị cho tham số đầu tiên (LEADING, TRAILING, BOTH), hàm TRIM sẽ mặc định là BOTH và loại bỏ trim\_character khỏi cả mặt trước và cuối chuỗi.
- Nếu bạn không chỉ định trim\_character, hàm TRIM sẽ mặc định ký tự sẽ bị xóa dưới dạng ký tự khoảng trắng.

Hàm LTRIM( string ) dùng để loại bỏ tất cả các ký tự khoảng trắng từ phía bên trái của chuỗi.

Hàm RTRIM( string ) dùng để loại bỏ tất cả các ký tự khoảng trắng từ phía bên phải của chuỗi.

Ví dụ:

```
SELECT TRIM(LEADING ' ' FROM 'Hello MySQL '); -- Kết quả: 'Hello MySQL '
SELECT TRIM(TRAILING ' ' FROM ' Hello MySQL '); -- Kết quả: ' Hello MySQL'
SELECT TRIM(BOTH ' ' FROM ' Hello MySQL '); -- Kết quả: 'Hello MySQL'
SELECT TRIM(' ' FROM ' Hello MySQL '); -- Kết quả: 'Hello MySQL'
SELECT TRIM(' Hello MySQL '); -- Kết quả: 'Hello MySQL'

SELECT TRIM(LEADING '0' FROM '0001233456790');-- Kết quả: '1233456790'
SELECT TRIM(TRAILING '0' FROM '0001233456790');-- Kết quả: '000123345679'
SELECT TRIM(BOTH '0' FROM '0001233456790');-- Kết quả: '123345679'

SELECT LTRIM(' Hello MySQL '); -- Kết quả: 'Hello MySQL '
SELECT RTRIM(' Hello MySQL '); -- Kết quả: ' Hello MySQL'
```

## 7.17 Hàm INSERT

Hàm INSERT dùng để chèn một chuỗi con vào một chuỗi tại một vị trí đã chỉ định cho một số ký tự nhất định.

Cú pháp: `INSERT(string, position, number, substring)`

Trong đó:

- `string`: Chuỗi để sửa đổi.
- `position`: Vị trí trong chuỗi để chèn chuỗi con.
- `number`: Số lượng ký tự để thay thế trong chuỗi.
- `substring`: Chuỗi con để chèn vào chuỗi.

Lưu ý:

- Vị trí đầu tiên trong `string` là 1
- Nếu `position` không nằm trong độ dài của `string`, hàm `INSERT` sẽ trả về `string`
- Nếu `number` không nằm trong độ dài của phần còn lại của chuỗi, hàm `INSERT` sẽ thay thế `string` bắt đầu từ `position` cho đến hết `string`

Ví dụ:

```
SELECT INSERT('Hello World', 7, 5, 'MySQL'); -- Kết quả: 'Hello MySQL'
SELECT INSERT('abcgh', 4, 0, 'def'); -- Kết quả: 'abcdefgh'
SELECT INSERT('abcgh', 4, 2, 'def'); -- Kết quả: 'abcdef'
SELECT INSERT('abcgh', 40, 2, 'def'); -- Kết quả: 'abcgh'
```

## 7.18 Hàm INSTR

Hàm `INSTR(string, substring)` dùng để trả về vị trí của chuỗi con trong chuỗi, hàm `INSTR` thực hiện tìm kiếm không phân biệt chữ hoa chữ thường. Nếu không tìm thấy chuỗi con trong chuỗi, thì hàm sẽ trả về 0.

Ví dụ:

```
SELECT INSTR('Hello MySQL', 'H'); -- Kết quả: 1
SELECT INSTR('Hello MySQL', 'h'); -- Kết quả: 1
SELECT INSTR('Hello MySQL', 'o'); -- Kết quả: 5
SELECT INSTR('Hello MySQL', 'l'); -- Kết quả: 3
SELECT INSTR('Hello MySQL', 'k'); -- Kết quả: 0
SELECT INSTR('Hello MySQL', 'MySQL'); -- Kết quả: 7
```

## 7.19 Hàm FIELD

Hàm FIELD(value, val1, val2, val3, ...) dùng để trả về vị trí của một giá trị trong danh sách các giá trị. Với value là giá trị cần tìm trong danh sách và val1, val2, val3, ... là danh sách các giá trị sẽ được tìm kiếm.

- Nếu không tìm thấy giá trị trong danh sách các giá trị, hàm FIELD sẽ trả về 0.
- Nếu giá trị là NULL, hàm FIELD sẽ trả về 0.
- Nếu tất cả các đối số trong hàm FIELD là các giá trị chuỗi, thì việc tìm kiếm được thực hiện dưới dạng các giá trị chuỗi.
- Nếu tất cả các đối số trong hàm FIELD là các giá trị số, thì việc tìm kiếm được thực hiện dưới dạng các giá trị số.

Ví dụ:

```
SELECT FIELD(20, 10, 20, 15, 40); -- Kết quả: 2
SELECT FIELD('c', 'a', 'b', 'c', 'd', 'e', 'f'); -- Kết quả: 3
SELECT FIELD('C', 'a', 'b', 'c', 'd', 'e', 'f'); -- Kết quả: 3
SELECT FIELD('k', 'a', 'b', 'c', 'd', 'e', 'f'); -- Kết quả: 0
SELECT FIELD(NULL, 'a', 'b', 'c', 'd', 'e', 'f'); -- Kết quả: 0
SELECT FIELD('b', NULL); -- Kết quả: 0
SELECT FIELD('a', ''); -- Kết quả: 0
```

## 7.20 Hàm FIND\_IN\_SET

Hàm FIND\_IN\_SET(string, string\_list) dùng để trả về vị trí của chuỗi trong danh sách chuỗi được phân tách bằng dấu phẩy.

- Nếu không tìm thấy string trong string\_list, hàm FIND\_IN\_SET sẽ trả về 0.
- Nếu string là NULL, hàm FIND\_IN\_SET sẽ trả về NULL.
- Nếu string\_list là một chuỗi rỗng, hàm FIND\_IN\_SET sẽ trả về 0.
- Nếu string\_list là NULL, hàm FIND\_IN\_SET sẽ trả về NULL

```
SELECT FIND_IN_SET(20, '10,20,15,40'); -- Kết quả: 2
SELECT FIND_IN_SET('c', 'a', 'b', 'c', 'd', 'e', 'f'); -- Kết quả: 3
SELECT FIND_IN_SET('C', 'a', 'b', 'c', 'd', 'e', 'f'); -- Kết quả: 3
SELECT FIND_IN_SET('k', 'a', 'b', 'c', 'd', 'e', 'f'); -- Kết quả: 0
SELECT FIND_IN_SET(NULL, 'a', 'b', 'c', 'd', 'e', 'f'); -- Kết quả: NULL
SELECT FIND_IN_SET('b', NULL); -- Kết quả: NULL
SELECT FIND_IN_SET('a', ''); -- Kết quả: 0
```

## 7.21 Hàm IFNULL, NULLIF

Chức năng của hàm IFNULL là dùng để kiểm tra một giá trị (hoặc một biểu thức giá trị)

- Nếu giá trị được kiểm tra là NULL thì hàm IFNULL sẽ trả về một giá trị được chỉ định.
- Nếu giá trị được kiểm tra không phải là NULL thì hàm IFNULL sẽ trả về chính giá trị đó.

Chức năng của hàm NULLIF là dùng để kiểm tra một giá trị (hoặc một biểu thức giá trị)

- Nếu hai giá trị đó giống nhau thì hàm NULLIF sẽ trả về giá trị NULL.
- Nếu hai giá trị đó khác nhau thì hàm NULLIF sẽ trả về giá trị thứ nhất.

Cú pháp:

IFNULL(giá trị cần kiểm tra, giá trị được trả về khi giá trị cần kiểm tra là NULL);  
NULLIF(giá trị thứ nhất, giá trị thứ hai);

```
SELECT IFNULL(NULL, 'Lap Trinh SQL'); -- Kết quả: 'Lap Trinh SQL'
SELECT IFNULL('Hello World', 'Lap Trinh SQL'); -- Kết quả: 'Hello World'
SELECT NULLIF('web', 'android'); -- Kết quả: 'web'
SELECT NULLIF('web', 'web'); -- Kết quả: NULL
```

Ví dụ: Cho bảng dữ liệu SINHVIEN sau đây:

MSSV	TEN	TUOI	DIACHI	GIOITINH
12150	Doan Du	28	HCM	Nam
12151	Vuong Ngu Yen	25	Long An	Nu
12152	NULL	26	Tay Ninh	Nu
12153	Ly Mac Sau	23	Binh Duong	Nu
12154	NULL	23	HCM	Nam

```
SELECT IFNULL(TEN, 'KHONG CO TEN') AS HOTEN FROM SINHVIEN;
```

Sau khi thực thi câu lệnh bên trên, ta được bảng dữ liệu sau:

HOTEN
Doan Du
Vuong Ngu Yen
KHONG CO TEN
Ly Mac Sau
KHONG CO TEN

## 7.22 Hàm ISNULL

Chức năng của hàm ISNULL là dùng để kiểm tra một giá trị (hoặc một biểu thức giá trị)

- Nếu giá trị được kiểm tra là NULL thì hàm ISNULL sẽ trả về 1.
- Nếu giá trị được kiểm tra không phải là NULL thì hàm ISNULL sẽ trả về 0.

Cú pháp: ISNULL(điều kiện cần kiểm tra)

Ví dụ:

```
SELECT ISNULL(NULL); -- Kết quả: 1
SELECT ISNULL('Lap Trinh SQL'); -- Kết quả: 0
SELECT ISNULL(TEN) FROM SINHVIEN; -- Kết quả: 1 1 0 1 0
```

## 7.23 Hàm IF

Chức năng của hàm IF là dùng để kiểm tra một biểu thức điều kiện, nếu nó đúng thì hàm IF sẽ trả về giá trị được chỉ định thứ nhất, nếu nó sai thì hàm IF sẽ trả về giá trị được chỉ định thứ hai.

Cú pháp: IF(biểu thức điều kiện, giá trị thứ nhất, giá trị thứ hai)

Ví dụ:

```
SELECT IF(1 < 2, 'DUNG', 'SAI'); -- Kết quả: 'DUNG'
SELECT IF(1 > 2, 'DUNG', 'SAI'); -- Kết quả: 'SAI'
SELECT IF(1<2, 1, 2); -- Kết quả: 1
```

Ví dụ:

```
SELECT TEN, IF(GIOITINH = 'Nam', 'Dan ong', 'Phu nu') AS GIOITINH FROM SINHVIEN;
```

Sau khi thực thi câu lệnh bên trên, ta được bảng dữ liệu sau:

TEN	GIOITINH
Doan Du	Dan ong
Vuong Ngu Yen	Phu nu
Nham Doanh Doanh	Phu nu
Ly Mac Sau	Phu nu
Au Duong Phong	Dan ong

# CHƯƠNG 8. SQL NÂNG CAO

## 8.1 Function và Store Procedure

Là đoạn chương trình kịch bản (programming scripts) với các Câu truy vấn sql nhúng (embedded SQL) được lưu dưới dạng đã được biên dịch và thi hành thực tiếp bởi một hệ quản trị cơ sở dữ liệu.

Hàm là một chương trình con (Sub Program) dùng để thực hiện một xử lý tính toán và trả về kết quả. Thủ tục cho phép lưu trữ các logic ứng dụng trên CSDL.

Trong MySQL, chúng ta sử dụng delimiter khi bắt đầu và kết thúc nội dung của một Function hoặc một Stored Procedure.

### Cú pháp tạo thủ tục/hàm

```
CREATE PROCEDURE [IF NOT EXISTS] ten_thu_tuc ([proc_parameter[,...]])  
[READS SQL DATA DETERMINISTIC]  
BEGIN  
    <Câu truy vấn sql>  
END  
  
CREATE FUNCTION [IF NOT EXISTS] ten_ham ([func_parameter[,...]])  
RETURNS data_type  
[READS SQL DATA DETERMINISTIC]  
BEGIN  
    <Câu truy vấn sql>  
END
```

Trong đó:

- + proc\_parameter: [ IN | OUT | INOUT ] param\_name data\_type |
- + func\_parameter: param\_name data\_type



Ví dụ: Câu lệnh sau tạo ra một thủ tục inDanhSachSV.

```
DELIMITER //  
CREATE PROCEDURE inDanhSachSV()  
BEGIN  
    SELECT * FROM SINHVIEN;  
END//
```

**Cú pháp gọi thủ tục/hàm**

```
CALL ten_thu_tuc;
SELECT ten_ham;
```



Ví dụ: Tạo thủ tục demSV dùng để đếm số lượng sinh viên như sau.

```
DELIMITER //
CREATE PROCEDURE demSV (IN gt VARCHAR(3), OUT soluong INT)
BEGIN
    SELECT COUNT(*) INTO soluong FROM SINHVIEN
    WHERE GIOITINH = gt;
END//

CALL demSV('Nam', @soluong);
SELECT @soluong; -- Kết quả: 2
```

Ví dụ: Viết hàm tìm tên sinh viên theo mã số sinh viên.

```
DELIMITER //
CREATE FUNCTION timSinhVien(ID INT)
RETURNS VARCHAR(20)
READS SQL DATA DETERMINISTIC
BEGIN
    DECLARE TEN_F VARCHAR(20);
    SET TEN_F = '';

    SELECT TEN INTO TEN_F FROM SINHVIEN WHERE MSSV = ID;
    RETURN TEN_F;
END; //
DELIMITER ;

SELECT timSinhVien(12151); -- Kết quả: Vuong Ngu Yen
```

**Cú pháp xóa thủ tục/hàm**

```
DROP PROCEDURE [ IF EXISTS ] ten_thu_tuc;
DROP FUNCTION [ IF EXISTS ] ten_ham;
```



**Các cú pháp hiển thị thông tin các thủ tục/hàm**

```
SHOW FUNCTION STATUS;
SHOW PROCEDURE STATUS;
SHOW FUNCTION STATUS LIKE 'tim%';
SHOW CREATE PROCEDURE sp_name;
```



Một số cấu trúc điều khiển dùng trong Hàm, thủ tục:

**Mệnh đề IF...THEN...**

```
IF condition THEN
    commands;
[ELSE IF condition THEN
    commands;]
[ELSE
    commands;]
END IF;
```



Ví dụ: Viết hàm cho biết sinh viên đang tìm đã đủ 18 tuổi hay chưa.

```
DELIMITER //
CREATE FUNCTION timTuoiSinhVien(ID INT)
RETURNS VARCHAR(50)
READS SQL DATA DETERMINISTIC
BEGIN
    DECLARE STT VARCHAR(20) DEFAULT STT=' ';
    DECLARE AGE INT DEFAULT AGE = 0;

    SELECT TUOI INTO AGE FROM SINHVIEN WHERE MSSV = ID;
    IF AGE < 18 THEN
        SET STT = 'Nho hon 18 tuoi';
    ELSE
        SET STT = 'Lon hon 18 tuoi';
    END IF;
    RETURN STT;
END; //
DELIMITER ;

SELECT timTuoiSinhVien(12150) AS KQ; -- Kết quả: Lon hon 18 tuoi
```

**Mệnh đề CASE**

CASE expression

```

WHEN value1 THEN commands;
[WHEN value2 THEN commands;]
[ELSE commands;]
END CASE;
```



Ví dụ: Viết hàm cho biết sinh viên đang tìm là nam hay nữ.

```

DELIMITER //
CREATE FUNCTION timGioiTinhSinhVien(ID INT)
RETURNS VARCHAR(50)
READS SQL DATA DETERMINISTIC
BEGIN
    DECLARE STT VARCHAR(20);
    DECLARE GENDER VARCHAR(3);

    SET STT = '';
    SET GENDER = '';

    SELECT GIOITINH INTO GENDER FROM SINHVIEN WHERE MSSV = ID;
    CASE GENDER
        WHEN 'Nam' THEN
            SET STT = 'Gioi tinh la nam';
        WHEN 'Nu' THEN
            SET STT = 'Gioi tinh la nu';
        ELSE
            SET STT = 'Khong xac dinh';
    END CASE;
    RETURN STT;
END; //
DELIMITER ;

SELECT timGioiTinhSinhVien(12150) AS KQ; -- Kết quả: Gioi tinh la nam
```

**Mệnh đề WHILE**

```

WHILE condition DO
    commands;
END WHILE
```



Ví dụ: Viết hàm in ra màn hình các số từ a tới b.

```

DELIMITER //
CREATE FUNCTION chuoia(a INT, b INT)
RETURNS VARCHAR(255)
READS SQL DATA DETERMINISTIC
BEGIN
DECLARE str VARCHAR(255) DEFAULT '';
WHILE (a <= b) DO
SET str = CONCAT(str,a,' ');
SET a = a + 1;
END WHILE;

RETURN str;
END; //
DELIMITER ;

SELECT chuoia(1,5) AS KQ; -- Kết quả: 1,2,3,4,5,

```

Cách sử dụng biến trong Function và Store Procedure.

#### Cú pháp khai báo và gán giá trị cho biến

```

DECLARE var_name data_type DEFAULT [default_value];
SET var_name = value;

```



Ví dụ:

```

DECLARE AGE INT DEFAULT 0;
DECLARE TONGSV INT DEFAULT 0;
SET AGE = 18;

-- Gán giá trị thông qua lệnh SELECT
SELECT COUNT(*) INTO TONGSV FROM SINHVIEN

```

## 8.2 Trigger

Trigger là một đối tượng được định danh trong CSDL và được gắn chặt với một sự kiện xảy ra trên một bảng nào đó (điều này có nghĩa là nó sẽ được tự động thực thi khi xảy ra một sự kiện trên một bảng). Các sự kiện này bao gồm: INSERT, UPDATE, hoặc DELETE.

Trigger được dùng để kiểm tra dữ liệu, đồng bộ hóa dữ liệu, đảm bảo các mối quan hệ giữa các bảng, lưu lại thay đổi hoặc cập nhật dữ liệu các bảng khác.

Trigger thực hiện các cập nhật lên bảng dữ liệu vì thế nó làm tăng lượng công việc lên CSDL và làm cho hệ thống chạy chậm.

### Cú pháp tạo Trigger

```
CREATE TRIGGER trigger_name
trigger_time trigger_event
ON table_name FOR EACH ROW
BEGIN
    <Câu truy vấn sql>
END
```

Trong đó

trigger\_time: { BEFORE | AFTER }  
trigger\_event: { INSERT | UPDATE | DELETE }



Ví dụ: Tạo một trigger có tên them\_sinhvien. Bất cứ khi nào xảy ra việc thêm dữ liệu vào bảng SINHVIEN thì những thay đổi đó sẽ được ghi nhận và được đưa vào lưu ở một bảng khác.

```
-- Tạo bảng ghi nhận những thay đổi khi thêm sinh viên
CREATE TABLE MESSAGE(
    STT INT PRIMARY KEY AUTO_INCREMENT,
    SMS VARCHAR(50)
);

DELIMITER //
CREATE TRIGGER them_sinhvien
AFTER INSERT ON SINHVIEN FOR EACH ROW
BEGIN
    INSERT INTO MESSAGE(SMS) VALUES (concat('Them thanh cong sinh vien ',NEW.MSSV));
END
//
```

## 8.3 View

View là các truy vấn SELECT được lưu lại như là một table ảo và sau đó ta có thể truy vấn dữ liệu từ view như thực hiện với table thật.

### Cú pháp tạo View

```
CREATE VIEW view_name AS
<Câu truy vấn sql>
```



### Cú pháp đổi tên View

```
RENAME TABLE old_view_name
TO new_view_name;
```



### Cú pháp xóa View

```
DROP VIEW [IF EXISTS] view_name;
```



Ví dụ: Tao một view lưu trữ tổng điểm của mỗi sinh viên.

```
CREATE VIEW totalScorePerStudent AS
SELECT MSSV, SUM(DIEM) AS TONGDIEM
FROM SINHVIEN
GROUP BY MSSV
ORDER BY TONGDIEM DESC;

-- Hiển thị danh sách các TABLE và VIEW có trong CSDL.
SHOW [FULL] TABLES;

-- In danh sách tổng điểm của mỗi sinh viên
SELECT * FROM totalScorePerStudent;

-- Đổi tên view totalScorePerStudent thành myView
RENAME TABLE totalScorePerStudent TO myView;

-- Xóa myView
DROP VIEW IF EXISTS myView;
```

## 8.4 Transaction

Transaction là một nhóm các câu lệnh SQL. Nếu một transaction được thực hiện thành công, tất cả các thay đổi dữ liệu được thực hiện trong transaction được lưu vào cơ sở dữ liệu. Nếu một transaction bị lỗi và được rollback, thì tất cả các sửa đổi dữ liệu sẽ bị xóa (dữ liệu được khôi phục về trạng thái trước khi thực hiện transaction).

Các lệnh sau đây được sử dụng để xử lý transaction.

- BEGIN hoặc START TRANSACTION - để bắt đầu một transaction.
- COMMIT - để xác nhận và lưu các thay đổi trên Database.
- ROLLBACK - để khôi phục lại các thay đổi.
- SAVEPOINT - tạo ra các điểm trong transaction để ROLLBACK.

Các lệnh điều khiển transaction chỉ được sử dụng với các lệnh thao tác dữ liệu DML như: INSERT, UPDATE và DELETE.

Để bắt đầu một transaction, chúng ta có thể sử dụng câu lệnh BEGIN hoặc START TRANSACTION và có thể sử dụng SET TRANSACTION để chỉ định các đặc tính cho transaction. Ví dụ, có thể chỉ định một transaction chỉ được đọc hoặc đọc viết.

### Cú pháp tạo TRANSACTION

```
[ BEGIN | START TRANSACTION ];  
[SET TRANSACTION [ READ WRITE | READ ONLY ] ;]
```



### Lệnh COMMIT

Lệnh COMMIT được sử dụng để lưu tất cả các thay đổi gọi bởi transaction vào cơ sở dữ liệu kể từ lệnh COMMIT hoặc ROLLBACK cuối cùng.

### Cú pháp lệnh COMMIT

```
COMMIT;
```



Ví dụ: Xóa các sinh viên có tuổi là 23 và sử dụng lệnh COMMIT để lưu các thay đổi trong cơ sở dữ liệu.

```
START TRANSACTION;  
DELETE FROM SINHVIEN WHERE TUOI = 23;  
COMMIT;
```

Các sinh viên có tuổi là 23 sẽ bị xóa và câu lệnh SELECT \* FROM SINHVIEN sẽ cho kết quả sau:

MSSV	TEN	TUOI	DIACHI	GIOITINH
12150	Doan Du	28	HCM	Nam
12151	Vuong Ngu Yen	25	Long An	Nu
12152	Nham Doanh Doanh	26	Tay Ninh	Nu

## Lệnh ROLLBACK

Lệnh ROLLBACK được sử dụng để hoàn tác các transaction chưa được lưu vào cơ sở dữ liệu kể từ khi lệnh COMMIT hoặc ROLLBACK cuối cùng được phát hành.

### Cú pháp lệnh ROLLBACK

ROLLBACK;



Ví dụ: Xóa các sinh viên có tuổi là 23 và sử dụng lệnh ROLLBACK để **xóa các thay đổi** trong cơ sở dữ liệu.

START TRANSACTION;

DELETE FROM SINHVIEN WHERE TUOI = 23;

ROLLBACK;

Các sinh viên có tuổi là 23 sẽ bị xóa, nhưng hoạt động xóa sẽ không ảnh hưởng đến bảng SINHVIEN và câu lệnh SELECT \* FROM SINHVIEN sẽ cho kết quả sau:

MSSV	TEN	TUOI	DIACHI	GIOITINH
12150	Doan Du	28	HCM	Nam
12151	Vuong Ngu Yen	25	Long An	Nu
12152	Nham Doanh Doanh	26	Tay Ninh	Nu
12153	Ly Mac Sau	23	Binh Duong	Nu
12154	Au Duong Phong	23	HCM	Nam

## Lệnh SAVEPOINT

Lệnh SAVEPOINT là một điểm trong một transaction khi bạn có thể cuộn transaction trở lại một điểm nhất định mà không quay trở lại toàn bộ transaction.

**Cú pháp lệnh SAVEPOINT**

SAVEPOINT savePoint\_name;



Lệnh này chỉ phục vụ trong việc tạo ra SAVEPOINT trong số tất cả các câu lệnh transaction. Lệnh ROLLBACK được sử dụng để hoàn tác một nhóm các transaction.

**Cú pháp cuộn lại một SAVEPOINT**

ROLLBACK TO savePoint\_name;



Ví dụ: Xóa ba sinh viên khác nhau từ bảng SINHVIEN và tạo SAVEPOINT trước mỗi lần xóa để có thể khôi phục dữ liệu.

```
START TRANSACTION;
SAVEPOINT point1;
DELETE FROM SINHVIEN WHERE MSSV = 12150;
SAVEPOINT point2;
DELETE FROM SINHVIEN WHERE TUOI = 12151;
SAVEPOINT point3;
DELETE FROM SINHVIEN WHERE TUOI = 12152;
ROLLBACK TO point2;
```

Kết quả trả về của câu lệnh SELECT \* FROM SINHVIEN như sau:

MSSV	TEN	TUOI	DIACHI	GIOITINH
12151	Vuong Ngu Yen	25	Long An	Nu
12152	Nham Doanh Doanh	26	Tay Ninh	Nu
12153	Ly Mac Sau	23	Binh Duong	Nu
12154	Au Duong Phong	23	HCM	Nam

**Cú pháp xóa SAVEPOINT**

RELEASE SAVEPOINT savePoint\_name;



Lưu ý: Sau khi một SAVEPOINT bị xóa, chúng ta không thể sử dụng lệnh ROLLBACK để hoàn tác các transaction sau SAVEPOINT đã xóa.

## 8.5 Cursor

Trong các truy vấn T-SQL, như tại các Stored Procedure, ta có thể sử dụng các con trỏ CURSOR để duyệt qua dữ liệu. Ta hiểu CURSOR là một tập hợp kết quả truy vấn (các hàng), với CURSOR ta có thể duyệt qua từng hàng kết quả để thi hành những tác vụ phức tạp.

Ở một thời điểm, CURSOR có thể truy cập bởi một con trỏ đến một hàng của nó, bạn chỉ thể dịch chuyển con trỏ từ dòng này sang dòng khác.

Để sử dụng con trỏ, ta cần thực hiện theo các bước:

Bước 1: khai báo con trỏ, trỏ đến một tập dữ liệu (kết quả của SELECT) bằng lệnh DECLARE.

### Cú pháp khai báo con trỏ

```
DECLARE cursor_name CURSOR FOR <Câu truy vấn sql>
```



Ví dụ có một tập dữ liệu từ câu lệnh Select như sau:

```
DECLARE mCursor CURSOR FOR
SELECT MSSV, TEN FROM SINHVIEN
```

Bước 2: Khi bắt đầu quá trình đọc các dòng dữ liệu từ Cursor trên, thì phải mở con trỏ, thực hiện như sau:

```
OPEN mCursor
```

Khi Cursor được mở, con trỏ sẽ trỏ tới dòng đầu tiên của tập dữ liệu, lúc này có thể đọc nội dung dòng đó bằng lệnh FETCH.

Bước 3: Đọc dữ liệu sử dụng lệnh như sau:

```
FETCH NEXT FROM mCursor INTO @id, @name
```

Lệnh trên sẽ đọc nội dung dòng hiện tại, lưu vào biến @id và @name (vì dữ liệu trong Cursor này có 2 cột). Nếu đọc thành công thì dịch chuyển con trỏ tới dòng tiếp theo.

Để kiểm tra việc FETCH thành công thì kiểm tra điều kiện @@FETCH\_STATUS = 0.

Bước 4: sau khi không còn dùng đến Cursor, cần đóng lại và giải phóng các tài nguyên nó chiếm giữ

```
CLOSE mCursor
DEALLOCATE mCursor
```

Các câu lệnh sử dụng Cursor ở trên, tổng hợp lại trong một ví dụ hoàn chỉnh như sau:

```
--Khai báo biến @id, @name để lưu nội dung đọc
DECLARE @id int
DECLARE @name nvarchar(200)

-- khai báo con trỏ mCursor
DECLARE mCursor CURSOR FOR
SELECT MSSV, TEN FROM SINHVIEN -- dữ liệu trả về

-- Mở con trỏ
OPEN mCursor

-- Đọc dòng đầu tiên
FETCH NEXT FROM mCursor
INTO @id, @name

--vòng lặp WHILE khi đọc Cursor thành công
WHILE @@FETCH_STATUS = 0
BEGIN
    --In kết quả hoặc thực hiện bất kỳ truy vấn nào dựa trên kết quả đọc được
    PRINT 'ID:' + CAST(@id as nvarchar)
    PRINT 'TITLE:' @title

    FETCH NEXT FROM mCursor -- Đọc dòng tiếp
    INTO @id, @title
END

CLOSE mCursor          -- Đóng Cursor
DEALLOCATE mCursor    -- Giải phóng tài nguyên
```

## CHƯƠNG 9. BÀI TẬP SQL

1. Hãy thực hiện các yêu cầu của bài tập 1-9 Chương 2 bằng ngôn ngữ SQL.

2. Cho lược đồ CSDL sau:

GIAOVIEN(MAGV,HOTEN,LUONG,PHAI,NGAYSINH,DIACHI, GVQL,MABM)

GIAOVIEN\_DT(MAGV,SDT)

BOMON(MABM,TENBM,PHONG,SDT,MAKHOA,TRUONGBM,NGAYNHANCHUC)

KHOA(MAKHOA,TENKHOA,NAMTL,PHONG,SDT,TRUONGKHOA,NGAYNHANCHUC)

DETAI(MADT,TENDT,KINHPHI,CAPQL,NGAYBD,NGAYKT,MACD,GVCNDT)

CONGVIEC(STT,MADT,TENCV,NGAYBD,NGAYKT)

CHUDE(MACD,TENCD)

THAMGIADT(MAGV,MADT,STT,PHUCAP,KETQUA)

Hãy thực hiện các yêu cầu sau bằng ngôn ngữ SQL:

1. Truy xuất tất cả thông tin của bảng giáo viên.
2. Lấy ra danh sách mã giáo viên, họ tên giáo viên và lương của giáo viên đó.
3. Lấy ra danh sách mã khoa và tên khoa tương ứng
4. Cho biết các khoa có năm thành lập từ 1980 trở về sau.
5. Cho biết mã khoa, tên khoa và tên giáo viên làm trưởng khoa đó.
6. Cho biết mã giáo viên, họ tên giáo viên và tên khoa của giáo viên đó đang làm việc.
7. Cho biết mã giáo viên, họ tên giáo viên và họ tên người quản lý chuyên môn giáo viên đó.
8. Cho biết thông tin lương của giáo viên (yêu cầu loại bỏ giá trị trùng).
9. Cho biết mã giáo viên, họ tên và lương của họ sau khi tăng lên 10%.
10. Cho biết mã giáo viên, họ tên giáo viên thuộc bộ môn “Hệ thống thông tin”
11. Cho biết mã giáo viên, họ tên giáo viên và họ tên người quản lý chuyên môn giáo viên.
12. Cho biết danh sách các giáo viên có tham gia đề tài.
13. Cho biết danh sách các giáo viên là chủ nhiệm đề tài.
14. Cho biết thông tin các giáo viên có họ là “Trần”.
15. Cho biết thông tin các giáo viên có địa chỉ ở “HCM”.
16. Cho biết thông tin các giáo viên có chữ “h” trong họ tên.
17. Cho biết thông tin các giáo viên có lương nằm trong khoảng từ 2000 đến 2500.
18. Cho biết thông tin các giáo viên có lương không nằm trong khoảng từ 2000 đến 2500.
19. Cho biết thông tin các giáo viên có độ tuổi từ 25 đến 35 tuổi.
20. Cho biết thông tin các khoa thành lập từ năm 1980 đến năm 1990.
21. Cho biết thông tin các giáo viên có mức phụ cấp tham gia đề tài từ 1.5 đến 2.0
22. Cho biết thông tin các trưởng bộ môn nhận chức từ đầu năm 2015 đến hết năm 2016.
23. Cho biết mã giáo viên, họ tên và lương. Kết quả trả về sắp xếp mức lương tăng dần.

- 
- 24. Cho biết mã giáo viên, họ tên và lương. Kết quả trả về sắp xếp mức lương giảm dần, sau đó các kết quả cùng mức lương xếp theo mã giáo viên giảm dần.
  - 25. Cho biết mã đề tài, số thứ tự công việc, tên công việc. Kết quả trả về sắp xếp mã đề tài tăng dần, số thứ tự công việc giảm dần.
  - 26. Cho biết mã giáo viên của những giáo viên có lương  $\geq 2500$  hoặc có người thân là nam.
  - 27. Cho biết mã đề tài có giáo viên tham gia với họ là “Trần” hoặc người chủ nhiệm đề tài có họ là “Trần”.
  - 28. Tính tổng lương, lương lớn nhất, lương nhỏ nhất và lương trung bình của toàn bộ giáo viên.
  - 29. Tính tổng lương, lương lớn nhất, lương nhỏ nhất và lương trung bình của giáo viên thuộc bộ môn “Hệ thống thông tin”.
  - 30. Cho biết số lượng giáo viên ở bộ môn “Hệ thống thông tin”.
  - 31. Cho biết lương trung bình của giáo viên từng bộ môn.
  - 32. Cho biết số lượng giáo viên tham gia cho mỗi đề tài.
  - 33. Xuất ra tên giáo viên và số lượng đề tài giáo viên đó đã làm.
  - 34. Xuất ra tên khoa có số lượng giáo viên trong khoa là nhiều nhất.
  - 35. Cho biết bộ môn nào có lương trung bình hơn 2000.
  - 36. Xuất ra số lượng giáo viên trong từng bộ môn mà số giáo viên  $> 1$ .
  - 37. Cho biết đề tài nào có số lượng giáo viên tham gia trên 3 giáo viên.
  - 38. Xuất ra danh sách giáo viên có mức lương  $>$  mức lương trung bình của tất cả giáo viên.
  - 39. Xuất ra thông tin của khoa có nhiều hơn 2 giáo viên.
  - 40. Xuất ra danh sách giáo viên tham gia nhiều hơn 1 đề tài.
  - 41. Tìm những trưởng bộ môn có tham gia ít nhất 1 đề tài.
  - 42. Tìm những giáo viên có lương lớn hơn của ít nhất 1 giáo viên thuộc bộ môn “Hệ thống thông tin”.
  - 43. Tìm những giáo viên có lương lớn hơn tất cả giáo viên thuộc bộ môn “Hệ thống thông tin”.
  - 44. Cho biết mã giáo viên, họ tên giáo viên chưa tham gia đề tài nào.
  - 45. Cho biết danh sách giáo viên gồm mã, họ tên, phái, ngày sinh.
  - 46. Cho biết danh sách đề tài gồm mã đề tài, tên đề tài, kinh phí.
  - 47. Cho biết danh sách giáo viên có lương  $> 1800$ .
  - 48. Cho biết danh sách giáo viên thuộc bộ môn có mã là HTTT.
  - 49. Cho biết danh sách giáo viên thuộc bộ môn mã là HTTT và có lương  $> 2000$ .
  - 50. Cho biết những bộ môn chưa biết người làm trưởng bộ môn.
  - 51. Cho biết những bộ môn đã phân công giáo viên làm trưởng bộ môn.
  - 52. Cho biết danh sách gồm mã, họ tên, phái, ngày sinh của các giáo viên có lương lớn hơn 2000.
  - 53. Cho biết danh sách gồm mã các giáo viên vừa tham gia đề tài mã số 001 vừa tham gia đề tài có mã là 002.
  - 54. Cho biết danh sách gồm mã các giáo viên có tham gia đề tài mã số 001 nhưng không tham gia đề tài mã số 002.
  - 55. Cho biết danh sách gồm mã các giáo viên có tham gia đề tài mã số 001 hoặc mã số 002.

- 
56. Cho biết các giáo viên thuộc bộ môn HTTT tham gia tất cả các công việc của các đề tài cấp trường. Danh sách kết xuất gồm mã giáo viên, mã đề tài, số thứ tự.
  57. Liệt kê danh sách các thể hiện cho biết các giáo viên thuộc bộ môn mã là MMT tham gia tất cả các công việc liên quan đến đề tài 001.
  58. Liệt kê danh sách các thể hiện cho biết các giáo viên thuộc bộ môn tên là “Mạng máy tính” tham gia tất cả các công việc liên quan đến đề tài tên là “Ứng dụng hóa học xanh”.
  59. Liệt kê danh sách các thể hiện cho biết giáo viên mà là 003 tham gia tất cả các công việc liên quan đến đề tài có mã là 001.
  60. Cho biết danh sách các giáo viên và mã, tên bộ môn mà giáo viên trực thuộc. Danh sách kết xuất gồm MAGV, HOTEN, PHAI, NGAYSINH, MABM, TENBM.
  61. Cho biết danh sách các bộ môn và tên của người làm trưởng bộ môn.
  62. Cho biết danh sách các gồm thông tin các bộ môn và tên của người làm trưởng bộ môn, đối với những bộ môn chưa biết giáo viên nào làm trưởng bộ môn thì tại các cột cho biết mã và tên của trưởng bộ môn mang giá trị rỗng (null).
  63. Cho biết danh sách gồm thông tin giáo viên và đề tài mà giáo viên đã tham gia, những giáo viên chưa có tham gia đề tài thì tại các cột cho biết thông tin đề tài hiện giá trị rỗng.
  64. Cho biết danh sách gồm mã, họ tên, phái, ngày sinh của các giáo viên thuộc bộ môn tên là “Hệ thống thông tin”.
  65. Với nhiều đề tài cấp trường và cấp Đại học quốc gia thuộc chủ đề là “Quản lý giáo dục”, cho biết mã và tên giáo viên làm nhiệm đề tài.
  66. Cho biết danh sách giáo viên và tên người quản lý chuyên môn với kết quả gồm các cột sau: MAGV, HOTEN, NGAYSINH, TEN\_GVQLCM. Chỉ xuất thông tin các giáo viên có người quản lý chuyên môn.
  67. Cho biết danh sách gồm mã và tên giáo viên có tham gia đề tài tên là “HTTT quản lý các trường ĐH” hoặc đề tài có tên là “HTTT quản lý giáo vụ cho một Khoa”.
  68. Cho biết danh sách gồm mã và tên các giáo viên vừa có tham gia đề tài tên là “Ứng dụng hóa học xanh” vừa có tham gia đề tài có tên là “Nghiên cứu tế bào gốc”.
  69. Những giáo viên nào chưa từng tham gia đề tài (mã giáo viên, tên giáo viên).
  70. Cho biết các giáo viên có người quản lý chuyên môn không ở cùng một thành phố.
  71. Cho biết danh sách các giáo viên tham gia tất cả các công việc của đề tài mã là 001.
  72. Có tất cả bao nhiêu giáo viên.
  73. Mỗi bộ môn có bao nhiêu giáo viên (Mã bộ môn, tên bộ môn, số giáo viên).
  74. Mỗi bộ môn có bao nhiêu giáo viên sinh trước năm 1985.
  75. Cho biết những bộ môn có số giáo viên nữ lớn hơn 5.
  76. Có bao nhiêu đề tài được thực hiện từ năm 2020 đến năm 2022.
  77. Thêm vào bảng THAMGIADT các bộ dữ liệu cho biết giáo viên mã là 003 tham gia tất cả các công việc của đề tài mã là 001.
  78. Xóa các dòng dữ liệu liên quan đến đề tài 002 trong bảng THAMGIADT.
  79. Cập nhật lương của những giáo viên thuộc bộ môn mã là HTTT tăng 1.5 lần.
  80. Sửa phụ cấp cho những giáo viên tham gia đề tài mã là 006 thành 2.

---

3. Cho lược đồ CSDL sau (mô tả ở bài tập 2 Chương 3):

SINHVIEN(MSSV, HOTEN, PHAI, NGAYSINH, DIACHI, MANGANH)  
NGANH(MANGANH, TENNGANH, SOCD, TSSV)  
CHUYENDE (MACD, TENCD, SSVTD)  
CD\_NGANH(MACD, MANGANH)  
CD\_MO(MACD, HOCKY, NAM)  
DANGKY(MSSV, MACD, HOCKY, NAM, DIEM)

Hãy thực hiện các yêu cầu sau bằng ngôn ngữ SQL:

1. Liệt kê danh sách sinh viên gồm mã, họ tên, phái, ngày sinh.
2. Liệt kê danh sách sinh viên thuộc ngành tên là “Hệ thống thông tin” (MSSV, HOTEN, PHAI, NGAYINH).
3. Cho biết các ngành có tổng số sinh viên theo học từ trước đến nay lớn hơn 2000 (MANGANH, TENNGANH).
4. Những chuyên đề nào chỉ cho phép không quá 100 sinh viên đăng ký mỗi khi được mở (MACD, TENCD).
5. Danh sách các chuyên đề bắt buộc đối với ngành tên là “Mạng máy tính” (MACD, TENCD).
6. Mỗi chuyên đề có tất cả bao nhiêu ngành phải học (MACD, TENCD, SONGANH).
7. Danh sách các sinh viên đăng ký học một chuyên đề tên là “Java” nhiều hơn 1 lần.
8. Cho danh sách các sinh viên thuộc ngành tên là “Hệ thống thông tin” đã đăng ký học chuyên đề “Oracle” (MSSV, HOTEN).
9. Danh sách các ngành phải học nhiều hơn 2 chuyên đề (MANGANH, TENNGANH).
10. Cho danh sách các sinh viên đã đăng ký nhiều hơn 2 chuyên đề trong học kỳ 1 năm 2022.
11. Cho biết các ngành phải học chuyên đề “Java” hoặc chuyên đề “Oracle”.
12. Cho biết các ngành vừa phải học chuyên đề “Java” vừa phải học chuyên đề “Oracle”.
13. Cho biết các ngành phải học chuyên đề “Java” nhưng không phải học chuyên đề “Oracle”.
14. Liệt kê tên các chuyên đề mà sinh viên có mã là “0012345” đã học.
15. Danh sách các sinh viên đã đăng ký học 2 chuyên đề trong học kỳ 1 năm 2022.
16. Danh sách các sinh viên đã đăng ký học 2 chuyên đề trong học kỳ 1 năm 2022 đều có điểm là “Đạt”.
17. Cho danh sách các sinh viên đã học tất cả các chuyên đề bắt buộc đối với ngành ”Hệ thống thông tin”.
18. Danh sách các sinh viên đã đăng ký học nhiều hơn 1 chuyên đề trong năm học 2022.
19. Danh sách các sinh viên thuộc ngành “Hệ thống thông tin” đã học chuyên đề “Oracle” mà không học chuyên đề “CSDL phân tán” trong năm 2022.
20. Cho đến hiện tại, cho biết mỗi chuyên ngành có bao nhiêu sinh viên theo học.
21. Liệt kê các thể hiện dữ liệu cho biết tất cả các sinh viên thuộc ngành tên là “Hệ thống thông tin” đăng ký học tất cả các chuyên đề bắt buộc đối với ngành “Hệ thống thông tin” trong học kỳ 1 năm 2020 (MSSV, MACD, HOCKY, NAM).
22. Danh sách các sinh viên chưa học chuyên đề nào (MSSV, HOTEN).

- 
23. Cho biết năm nào, học kỳ nào mở tất cả các chuyên đề bắt buộc cho ngành “Hệ thống thông tin”.
24. Cho biết mã, tên của các chuyên đề thuộc chuyên ngành của sinh viên có mã là “0012345” mà sinh viên này chưa đăng ký học.
25. Danh sách các sinh viên thuộc ngành “Hệ thống thông tin” chỉ học duy nhất 1 chuyên đề trong học kỳ 1 năm 2022.
26. Cho biết tên các chuyên đề mà mọi ngành đều phải học chung.
27. Danh sách các chuyên đề bắt buộc đối với chuyên ngành tên là “Mạng máy tính” đã được mở ra trong học kỳ 1 năm 2023.
28. Danh sách các chuyên đề vừa là chuyên đề bắt buộc cho chuyên ngành “Hệ thống thông tin” vừa là chuyên đề bắt buộc cho chuyên ngành “Công nghệ tri thức”.
29. Cho danh sách các sinh viên chưa từng học lại một chuyên đề nào.
4. Cho lược đồ cơ sở dữ liệu Thư viện như sau:
- SACH(MASACH, TENSACH, TENNXB)  
SACH\_TACGIA(MASACH, TACGIA)  
NHAXUATBAN(TENNXB, DIACHI, SDT)  
SACH\_BANSAO(MASACH, MACHINHANH, SOLUONGBANSAO)  
NHANH\_THUVIEN(MACHINHANH, TENCHINHANH, DIACHI)  
SACH\_MUON(MASACH, MACHINHANH, SOTHE, NGAYMUON, NGAYTRA)  
NGUOIMUON(SOTHE, HOTEN, DIACHI, SDT)
- Hãy thực hiện các yêu cầu sau bằng ngôn ngữ SQL:
- (a). Có bao nhiêu bản sao của cuốn sách “The Lost Tribe” có trong nhánh thư viện có tên là “Shapstown”.
- (b). Có bao nhiêu bản sao của cuốn sách “The Lost Tribe” có trong mỗi nhánh thư viện.
- (c). Đưa ra tên của tất cả người mượn chưa mượn cuốn sách nào.
- (d). Với mỗi cuốn sách được mượn ra từ nhánh thư viện “Shapstown” có ngày trả là ngày hôm nay, hãy đưa ra Tên sách, Tên người mượn và địa chỉ người mượn.
- (e). Với mỗi thư viện nhánh, hãy đưa ra tên nhánh thư viện và tổng số sách được mượn ra từ nhánh này.
- (f). Đưa ra tên, địa chỉ và số các sách do người này mượn với những người mượn nhiều hơn 5 cuốn sách.
- (g). Với mỗi cuốn sách có tác giả (hoặc đồng tác giả) là “Stephen King”, hãy đưa ra tên sách và số lượng các bản sao có tại chi nhánh thư viện có tên là “Central”.
5. Cho lược đồ CSDL sau:

THUYTHU(MATT, TENTT, BAC, TUOI)  
TAU (MATAU, TENTAU, MAUSAC)  
DANGKY(MATT, MATAU, NGAY)

---

Hãy thực hiện các yêu cầu sau bằng ngôn ngữ SQL:

1. Tên của những thủy thủ đã đăng ký tàu mã là 103.
  2. Tên của những thủy thủ có bậc lớn hơn 7.
  3. Tên và tuổi của những thủy thủ có bậc lớn hơn 7.
  4. Tìm tên thủy thủ, tên tàu và ngày đăng ký của tất cả những lần đăng ký.
  5. Tên của những thủy thủ có đăng ký tàu màu đỏ.
  6. Tìm màu của những con tàu mà thủy thủ tên là Hùng đã đăng ký.
  7. Tên của những thủy thủ đã đăng ký ít nhất 1 con tàu.
  8. Tên của những thủy thủ đã có lần đăng ký con tàu màu đỏ hoặc tàu màu xanh
  9. Tên của những thủy thủ đã có lần đăng ký con tàu màu đỏ lần con tàu màu xanh.
  10. Tên của những thủy thủ đã đăng ký tối thiểu 2 con tàu.
  11. Tìm mã của những thủy thủ có tuổi lớn hơn 20 chưa từng đăng ký con tàu màu đỏ.
  12. Tên của những thủy thủ đã đăng ký tất cả những con tàu có tên là “Marine”.
6. Cho cơ sở dữ liệu Công ty gồm các lược đồ:
- NHANVIEN(MANV, HO, TEN, NGAYSINH, GIOITINH, DIACHI, LUONG, NGS, MADV)  
DONVI(MADV, TENDV, NQL, NGAYBD)  
DUAN(MADA, TENDA, DIAIEM, MADV)  
PHUTHUOC(MANV, TENPT, NGAYSINH, GIOITINH, QUANHE)  
NHANVIEN\_DUAN(MANV, MADA, SOGIO)  
DONVI\_DIAIEM(MADV, DIAIEM)
- Hãy thực hiện các yêu cầu sau bằng ngôn ngữ SQL:
- (a). Đưa ra tên và địa chỉ của tất cả các nhân viên làm việc cho đơn vị nghiên cứu.
  - (b). Với mỗi dự án có địa điểm tại HCM, hãy liệt kê mã số dự án, mã số của đơn vị kiểm soát, tên, địa chỉ và ngày sinh của người quản lý đơn vị.
  - (c). Tìm tên của các nhân viên làm việc ở các dự án do đơn vị có mã số 5 kiểm soát.
  - (d). Tạo ra một danh sách các mã số dự án đối với các dự án có một nhân viên hoặc một người quản lý đơn vị kiểm soát dự án có tên là ‘Nam’.
  - (e). Đưa ra tên của tất cả các nhân viên có nhiều hơn hoặc bằng 2 người phụ thuộc.
  - (f). Đưa ra các nhân viên không có người phụ thuộc.
  - (g). Đưa ra tên của những người quản lý có ít nhất là một người phụ thuộc.
7. Cho lược đồ cơ sở dữ liệu quan hệ “Quản lý món ăn” như sau:

**NGLIEU**(MaNL, TenNL, CaloriNL, ProteinNL)

*Tân từ:* mô tả thông tin về nguyên liệu để chế biến món ăn. Mỗi nguyên liệu có một mã số duy nhất, tên và thông tin về các thành phần dinh dưỡng như Calori, Protein của nguyên liệu. Mỗi nguyên liệu có thể dùng chế biến nhiều món ăn.

---

### **MONAN**(MaMA, TenMA, MaLoai, Gia, CaloriMA, ProteinMA)

*Tân từ:* mô tả thông tin về món ăn. Mỗi món ăn có một mã số duy nhất, tên, giá thành và thông tin về các thành phần dinh dưỡng như Calori, Protein của món ăn. Mỗi món ăn thuộc một loại xác định.

### **LOAIMONAN**(MaLoai, TenLoai)

*Tân từ:* mô tả thông tin các loại món ăn. Mỗi loại món ăn có một mã số duy nhất và tên, ví dụ tên loại món ăn là: Món Canh, Món Mặn, Món Tráng Miệng, Cơm...

### **TPMONAN**(MaMA, MaNL, TLuong)

*Tân từ:* cho biết thành phần các nguyên liệu để chế biến món ăn. Mỗi thành phần nguyên liệu có trọng lượng tương ứng.

**Yêu cầu 1:** Tìm các khóa chính, khóa ngoại (nếu có) cho mỗi lược đồ quan hệ trên. Lưu ý, đối với khóa ngoại cần chỉ rõ tham chiếu đến khóa chính của lược đồ quan hệ nào.

**Yêu cầu 2:** Hãy thực hiện các yêu cầu sau bằng ngôn ngữ SQL:

- (a). Hãy cho biết những món ăn (MaMA, TenMA) có lượng calori (CaloriMA) lớn hơn 500.
- (b). Hãy cho biết những nguyên liệu (MaNL, TenNL) được sử dụng để chế biến món ăn có mã món ăn là “CA05” với trọng lượng lớn hơn 100.
- (c). Liệt kê các món ăn và thành phần nguyên liệu để chế biến món ăn đó (nếu có). Thông tin hiển thị gồm: MaMA, TenMA, MaNL.
- (d). Với mỗi loại món ăn (MaLoai), cho biết lượng calori (CaloriMA) trung bình của các món ăn thuộc loại món ăn đó.
- (e). Cho biết những nguyên liệu (MaNL, TenNL) chưa được dùng để chế biến món ăn nào.
- (f). Tìm các món ăn (MaMA) mà thành phần gồm tất cả các nguyên liệu có lượng protein (ProteinNL) từ 5 trở lên.
- (g). Hãy cho biết những nguyên liệu (MaNL, TenNL) có lượng protein (ProteinNL) < 20.
- (h). Hãy cho biết những món ăn (MaMA, TenMA) thuộc loại món ăn có tên là “Món Canh” và có giá lớn hơn 20000.
- (i). Liệt kê các nguyên liệu và món ăn dùng nguyên liệu đó để chế biến (nếu có). Thông tin hiển thị gồm: MaNL, TenNL, MaMA
- (j). Với mỗi loại món ăn (MaLoai), cho biết lượng protein (ProteinMA) cao nhất của các món ăn thuộc loại món ăn đó.
- (k). Cho biết những món ăn (MaMA, TenMA) có thành phần nguyên liệu để chế biến từ thịt bò (MaNL = “TB”) và thịt gà (MaNL = “TG”).
- (l). Tìm các món ăn (MaMA) mà thành phần gồm tất cả các nguyên liệu có lượng calori (CaloriNL) từ 50 trở lên.

---

8. Cho lược đồ cơ sở dữ liệu quan hệ “Quản lý các Tour du lịch” như sau:

**TOUR**(MaTour, TenTour, SoNgay, SoDem, PTDi, PTVe, GiaLe, GiaNhom)

*Tân từ:* mỗi tuyến du lịch cần lưu trữ mã Tour (MaTour), tên Tour (TenTour), bao nhiêu ngày (SoNgay), bao nhiêu đêm (SoDem), phương tiện đi (PTDi), phương tiện về (PTVe), giá lẻ(GiaLe) dành cho khách đăng ký < 5 người và giá nhóm(GiaNhom) dành cho khách đăng ký  $\geq 5$  người.

**TINHTP**(MaTTP, TenTTP, Mien)

*Tân từ:* lưu trữ tất cả các tỉnh (thành phố) trong nước, gồm có những thông tin: mã tỉnh (hoặc thành phố) (MaTTP), tên tỉnh thành phố (TenTTP) và thuộc miền (Mien) nào.

**DIEMDL**(MaDDL, TenDDL, MaTTP, Dactrung)

*Tân từ:* mỗi điểm du lịch cần lưu trữ mã điểm du lịch (MaDDL), tên điểm du lịch (TenDDL) và điểm này thuộc tỉnh (thành phố) nào (MaTTP), đặc trưng (Dactrung) của điểm du lịch này là gì ? (chỉ lưu 1 đặc trưng chính như: “Tắm biển”, “Leo núi”, “Mua sắm”, “Tham quan”...).

**CHITIET** (MaTour, MaDDL, Ngay, Dem)

*Tân từ:* ghi nhận mỗi Tour (MaTour) đi qua các điểm du lịch (MaDDL) nào, bao nhiêu ngày (Ngay), bao nhiêu đêm (Dem) của mỗi địa điểm du lịch.

**Yêu cầu 1:** Tìm các khóa chính, khóa ngoại (nếu có) cho mỗi lược đồ quan hệ trên. Lưu ý, đối với khóa ngoại cần chỉ rõ tham chiếu đến khóa chính của lược đồ quan hệ nào.

**Yêu cầu 2:** Hãy thực hiện các yêu cầu sau bằng ngôn ngữ SQL:

- (a). Liệt kê các Tour du lịch (Mã tour, tên tour, phương tiện đi, phương tiện về, giá nhóm) trong 2 ngày, 1 đêm và có giá nhóm dưới 2 triệu đồng.
- (b). Cho biết danh sách các tour (MaTour, TenTour) đi bằng “Máy bay”, về bằng “Ô tô” và đi qua các điểm du lịch có đặc trưng là “Tắm biển”. Sắp xếp theo thứ tự giá lẻ tăng dần.
- (c). Cho biết tên tỉnh/thành phố có nhiều điểm du lịch nhất.
- (d). Liệt kê tỉnh/thành phố (MaTTP, TenTTP) vừa có đặc trưng là “Tắm biển” vừa có đặc trưng là “Leo núi”.
- (e). Tìm địa điểm du lịch (MaDDL, TenDDL) mà tour nào cũng có đến.
- (f). Liệt kê các Tour du lịch (Mã tour, tên tour, số ngày, số đêm, giá lẻ) đi và về bằng máy bay có giá lẻ dưới 3 triệu đồng.
- (g). Cho biết những tuyến du lịch (MaTour, TenTour, SoNgay, SoDem, GiaLe) đi qua điểm du lịch có tên “Vịnh Hạ Long”, sắp xếp theo số ngày (giảm dần).
- (h). Cho biết tên tuyến du lịch (TenTour) đi qua nhiều điểm du lịch nhất.
- (i). Liệt kê tỉnh/thành phố có đặc trưng là “Tắm biển” mà không có đặc trưng là “Leo núi”.
- (j). Tìm các tour đi qua tất cả địa điểm du lịch ở miền Nam.