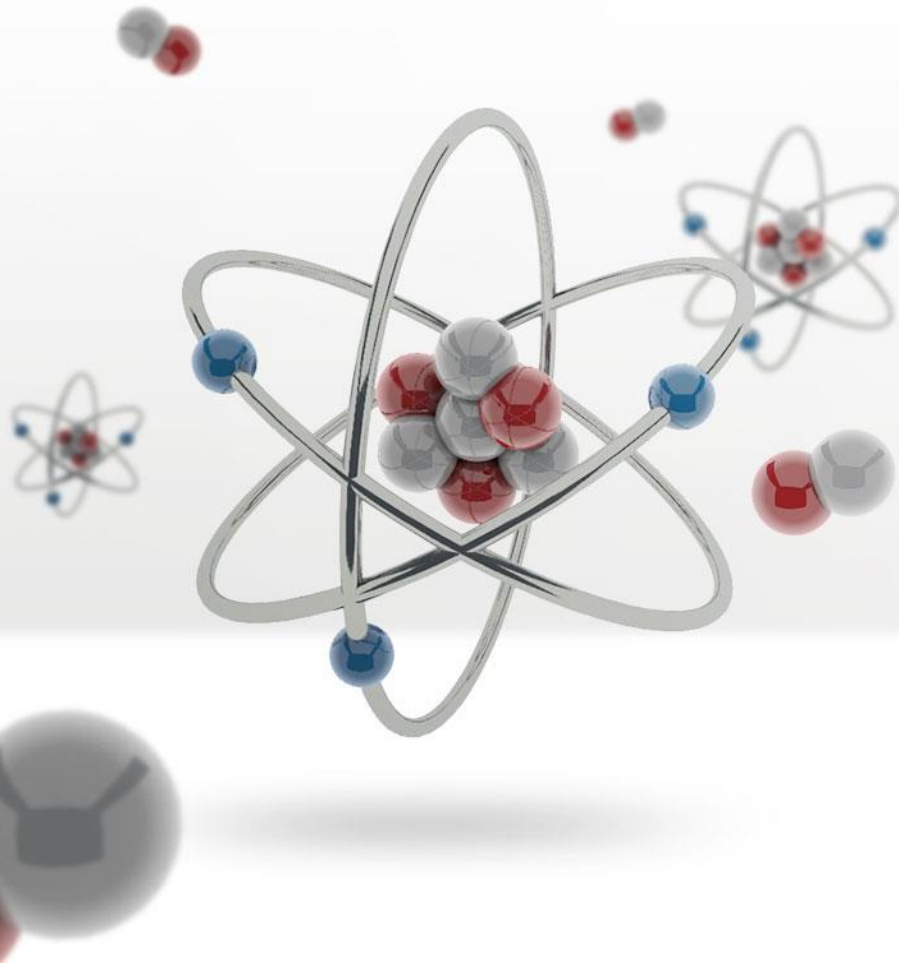




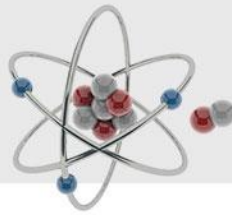
**CYBERSOFT**  
ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH



# CƠ SỞ DỮ LIỆU NỀN TẢNG

Hướng dẫn: Trương Tấn Khải

# TỔNG QUAN VỀ CSDL



1. Giới thiệu tổng quan về cơ sở dữ liệu
2. Các khái niệm cơ bản CSDL
3. Cài đặt hệ quản trị csdl mysql với workbend
4. Ngôn ngữ truy vấn SQL
  - *Ngôn ngữ định nghĩa dữ liệu (Tạo csdl, tạo bảng, sửa đổi cấu trúc bảng, xóa bảng)*
  - *Ngôn ngữ thao tác dữ liệu (Thêm, xóa, sửa, truy vấn lồng, gom nhóm và kết hợp)*
  - *Ngôn ngữ điều khiển dữ liệu (Cấp quyền, thu hồi quyền, từ chối quyền) (Tìm hiểu thêm sau)*

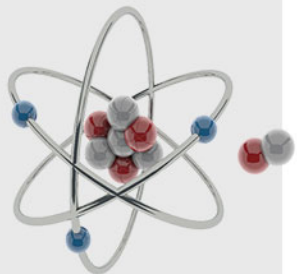


# 1. Giới thiệu tổng quan về cơ sở dữ liệu

## ❑ CSDL là gì ?

- Cơ sở dữ liệu là một hệ thống các thông tin có cấu trúc, được lưu trữ trên các thiết bị lưu trữ nhằm thỏa mãn yêu cầu khai thác thông tin đồng thời của nhiều người sử dụng hay nhiều chương trình ứng dụng chạy cùng một lúc với những mục đích khác nhau.
- Việc sử dụng hệ thống CSDL này sẽ khắc phục được những khuyết điểm của cách lưu trữ dưới dạng hệ thống tập tin, đó là:
  - Giảm trùng lặp thông tin ở mức thấp nhất, đảm bảo tính nhất quán và toàn vẹn dữ liệu.
  - Đảm bảo dữ liệu được truy xuất theo nhiều cách khác nhau, từ nhiều người khác nhau và nhiều ứng dụng khác nhau.
  - Tăng khả năng chia sẻ thông tin. Ví dụ nếu ta đặt hệ thống dữ liệu tại Việt Nam thì ở bên Mỹ nếu có password logi vào thì ta hoàn toàn có thể vào hệ thống để đọc tin.

Đương nhiên khi sử dụng các hệ thống CSDL thì bạn phải có một hệ quản trị CSDL. Hiện nay có rất nhiều hệ quản trị CSDL như MySQL, SQL SERVER, Oracle, MS Access. Trong loạt serie này ta sẽ tìm hiểu hệ quản trị CSDL MYSQL.



# 1. Giới thiệu tổng quan về cơ sở dữ liệu

## ❑ Hệ quản trị CSDL là gì?

➤ Hệ quản trị CSDL (Database Management System - DBMS) là các phần mềm giúp tạo các CSDL và cung cấp các cơ chế lưu trữ, truy cập theo các mô hình CSDL.

### ❑ Hệ quản trị CSDL phổ biến.

- MySQL.
- Oracle.
- SQL Server
- SQLite
- MongoDB
- PostgreSQL
- Redis



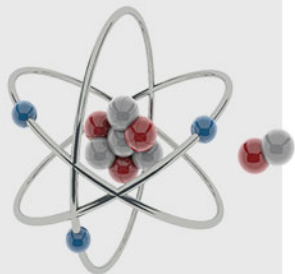
PostgreSQL



TURBODB  
Embedded



Microsoft  
SQL Server



## 2. Các khái niệm cơ bản trong CSDL

### ❑ Bảng là gì?

- **Bảng** là hình thức lưu trữ dữ liệu đơn giản và phổ biến nhất trong CSDL.
- **Bảng** về cơ bản là một bộ sưu tập các mục dữ liệu có quan hệ, nó bao gồm nhiều cột và hàng.

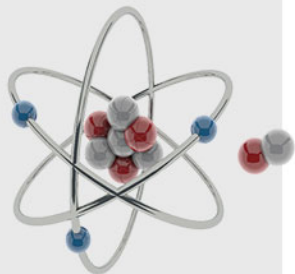
### ❑ Trường là gì?

- **Bảng** có thể được chia thành các mục nhỏ hơn gọi là trường (field).
- Ví dụ bảng SAN\_PHAM sẽ bao gồm các trường ID, TEN, SO\_LUONG, GIA.

	ID	TEN	SO_LUONG	GIA
▶	1	Điện thoại Nokia	20	10000000
	2	Laptop DELL	50	16000000
	3	Tủ lạnh SANYO	30	15000000
★	NULL	NULL	NULL	NULL

Trường

Bảng



## 2. Các khái niệm cơ bản trong CSDL

### ❑ Bản ghi là gì ?

- Một **bản ghi** còn được gọi là **hàng dữ liệu**, là từng **mục riêng lẻ** tồn tại trong **bảng**.
- Bản **ghi chính** là một **thực thể** nằm trong **bảng**.

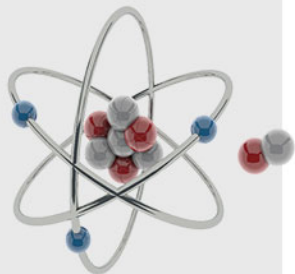
### ❑ Cột là gì ?

- **Cột** là một **thực thể** nằm **dọc** trong **bảng**, chứa tất cả **các thông tin** liên kết với một **trường** trong **bảng**.

	ID	TEN	SO_LUONG	GIA
▶	1	Điện thoại Nokia	20	10000000
	2	Laptop DELL	50	16000000
	3	Tủ lạnh SANYO	30	15000000
*	NULL	NULL	NULL	NULL

Bản ghi

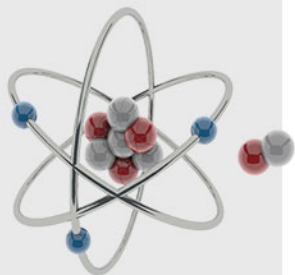
↑  
Cột



## 2. KHÁI NIỆM CB SQL - KIỂU DỮ LIỆU

### ❑ Kiểu văn bản

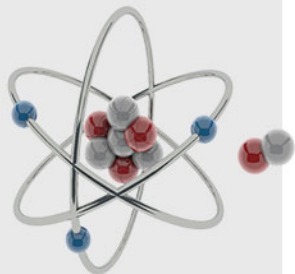
Kiểu dữ liệu	Mô tả
CHAR(size)	Dùng để lưu trữ một chuỗi ký tự có chiều dài tối đa là 255 ký tự
VARCHAR(size)	Dùng để lưu trữ một chuỗi ký tự có chiều dài tối đa là 255 ký tự. Nếu đặt "size" lớn hơn 255 thì nó sẽ chuyển sang kiểu TEXT
TINYTEXT	Dùng để lưu trữ một chuỗi ký tự có chiều dài tối đa là 255 ký tự
TEXT	Dùng để lưu trữ một chuỗi ký tự có chiều dài tối đa là 65,535 ký tự
BLOB	Dùng để lưu trữ dữ liệu nhị phân tối đa là 65,535 byte
MEDIUMTEXT	Dùng để lưu trữ một chuỗi ký tự có chiều dài tối đa là 16,777,215 ký tự
MEDIUMBLOB	Dùng để lưu trữ dữ liệu nhị phân tối đa là 16,777,215 byte
LONGTEXT	Dùng để lưu trữ một chuỗi ký tự có chiều dài tối đa là 4,294,967,295 ký tự
LOBLOB	Dùng để lưu trữ dữ liệu nhị phân tối đa là 4,294,967,295 byte



## 2. Các khái niệm cơ bản trong CSDL

### ❑ Kiểu ngày tháng

Kiểu dữ liệu	Mô tả
DATE()	Lưu trữ một ngày theo định dạng YYYY-MM-DD (Ví dụ: 2016-09-12 tức là lưu ngày 12 tháng 9 năm 2016)
TIME()	Lưu trữ thời gian theo định dạng HH:MI:SS (Ví dụ 17:25:36 tức là lưu 17 giờ 25 phút 36 giây)
YEAR()	Lưu trữ một năm theo định dạng hai số hoặc bốn số
DATETIME()	Lưu trữ một ngày cùng với thời gian theo định dạng YYYY-MM-DD HH:MI:SS (Ví dụ: 2016-09-12 17:25:36 tức là lưu ngày 12 tháng 9 năm 2016 lúc 17 giờ 25 phút 36 giây)

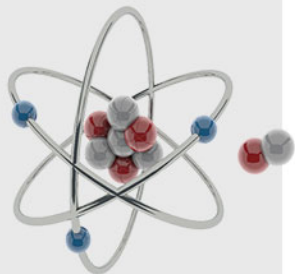




## 2. Các khái niệm cơ bản trong CSDL

### ❑ Kiểu ngày tháng

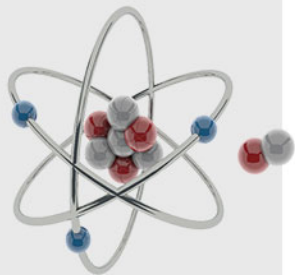
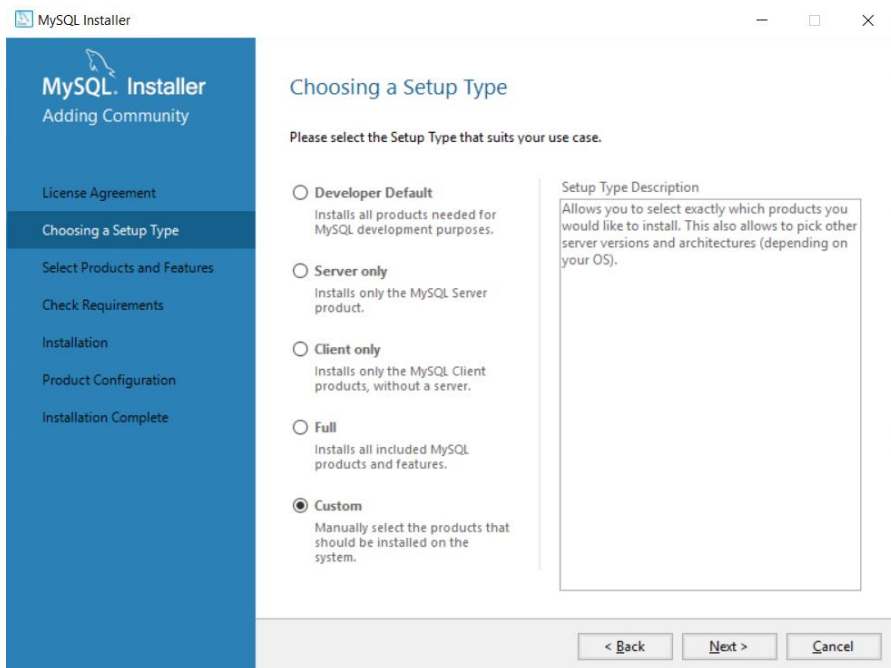
Kiểu dữ liệu	Mô tả
DATE()	Lưu trữ một ngày theo định dạng YYYY-MM-DD (Ví dụ: 2016-09-12 tức là lưu ngày 12 tháng 9 năm 2016)
TIME()	Lưu trữ thời gian theo định dạng HH:MI:SS (Ví dụ 17:25:36 tức là lưu 17 giờ 25 phút 36 giây)
YEAR()	Lưu trữ một năm theo định dạng hai số hoặc bốn số
DATETIME()	Lưu trữ một ngày cùng với thời gian theo định dạng YYYY-MM-DD HH:MI:SS (Ví dụ: 2016-09-12 17:25:36 tức là lưu ngày 12 tháng 9 năm 2016 lúc 17 giờ 25 phút 36 giây)



### 3. Hướng dẫn cài đặt MySQL với workbend

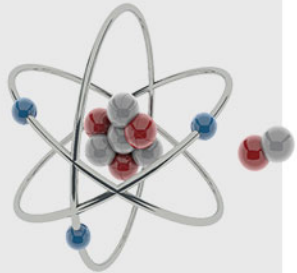
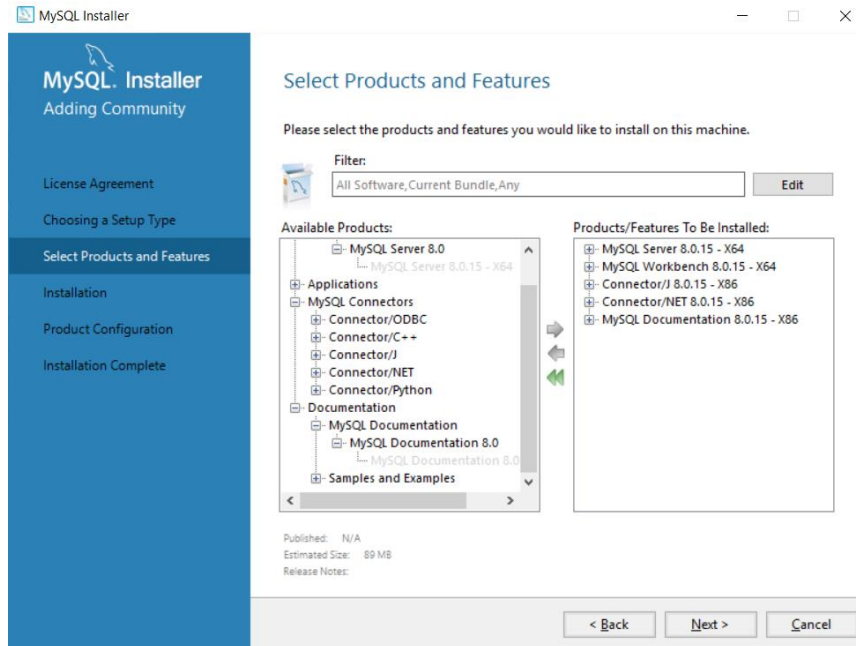
Link download: <https://dev.mysql.com/downloads/file/?id=484920>

**❑ Bước 1: Chọn các gói cần cài đặt, các bạn có thể chọn default**



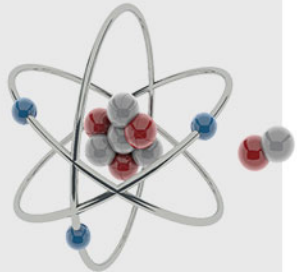
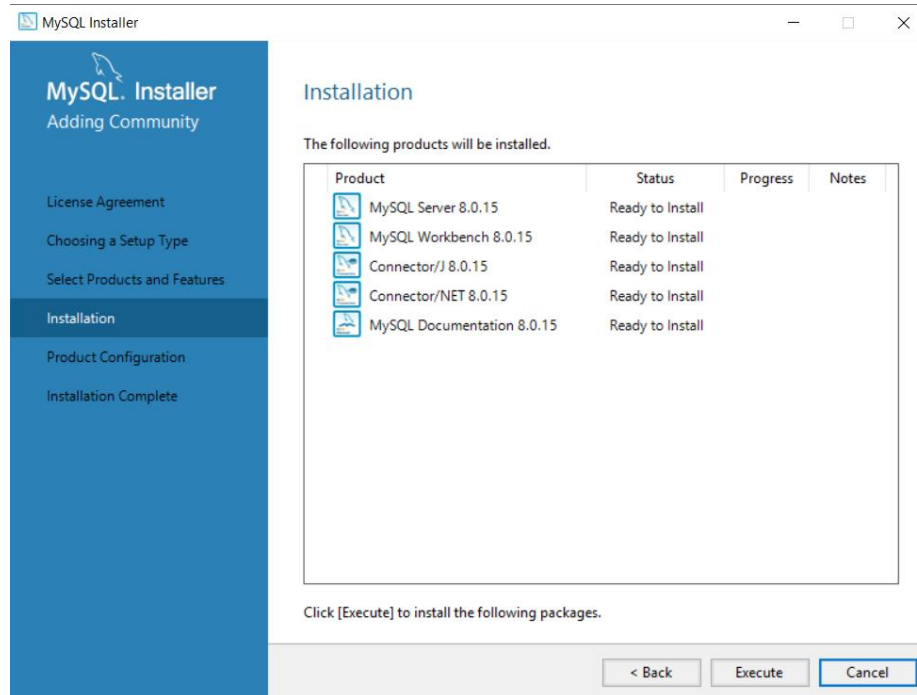
# 3. Hướng dẫn cài đặt MySQL với workbend

❑ **Bước 2: Nếu chọn custom thì các bạn sẽ chọn các gói sau để cài đặt cho mysql**



# 3. Hướng dẫn cài đặt MySQL với workbench

❑ **Bước 3: Tiến hành cài đặt và chọn Execute đợi tầm 5 phút.**



# 3. Hướng dẫn cài đặt MySQL với workbend

## ❑ Bước 4: Chọn các tùy chọn mặc định -> next

MySQL Installer

MySQL Server 8.0.15

Group Replication

Type and Networking

Authentication Method

Accounts and Roles

Windows Service

Apply Configuration

### Group Replication

☒ Standalone MySQL Server / Classic MySQL Replication  
Choose this option if you want to run the MySQL Server either standalone with the opportunity to later configure classic MySQL Replication.

Using this option you can manually configure your replication setup and provide your own high availability solution if required.

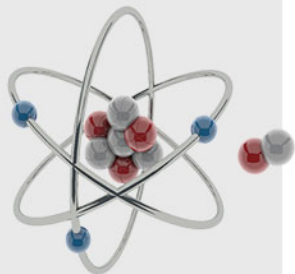
☐ Sandbox InnoDB Cluster Setup (for testing only)  
The [InnoDB cluster](#) technology provides an out-of-the-box HA (high availability) solution for MySQL using Group Replication technology.

This option allows you to test an InnoDB cluster setup on your local computer using several MySQL Server sandbox instances. Read more about this [here](#).

To setup a real-world production InnoDB cluster please choose the standard MySQL Server configuration instead on all desired hosts and use the MySQL Shell afterwards to create or expand the InnoDB cluster setup.

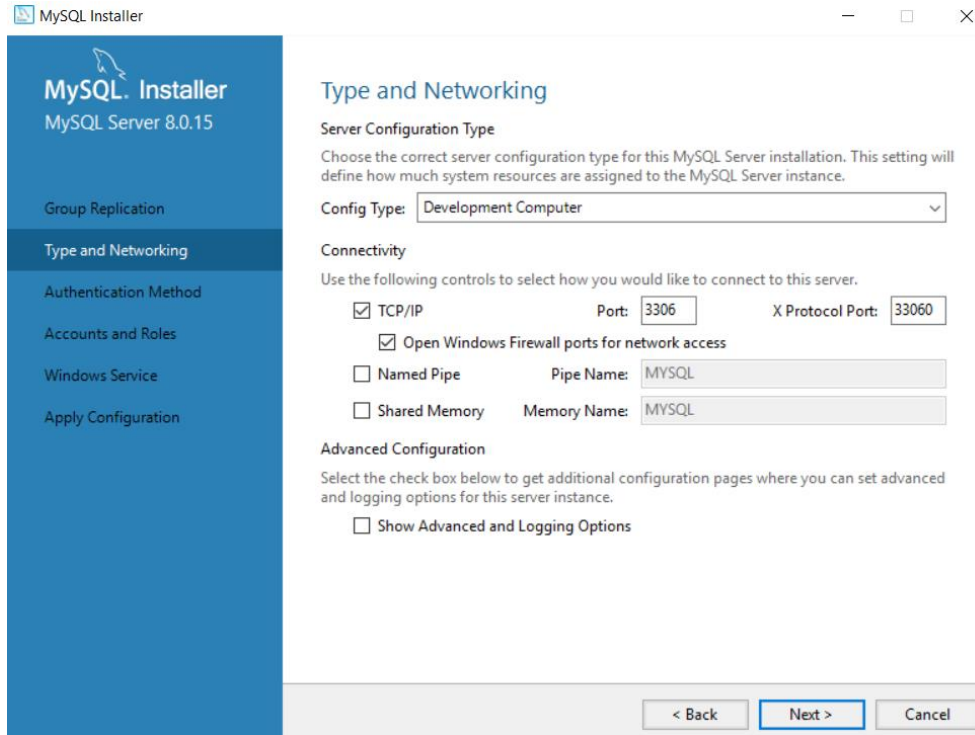
```
graph LR; ClientApp[Client App] <--> MySQLRouter[MySQL Router]; MySQLRouter <--> MySQLShell[MySQL Shell]; MySQLShell <--> InnoDBCluster((InnoDB Cluster)); InnoDBCluster <--> MySQLRouter;
```

Next > Cancel



# 3. Hướng dẫn cài đặt MySQL với workbend

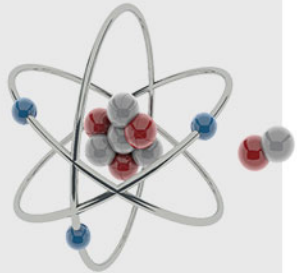
## ❑ Bước 5: Các thông số port ... để mặc định => next



The image shows the MySQL Installer window for MySQL Server 8.0.15. The left sidebar contains the following options: Group Replication, Type and Networking (selected), Authentication Method, Accounts and Roles, Windows Service, and Apply Configuration. The main area is titled 'Type and Networking' and contains the following sections:

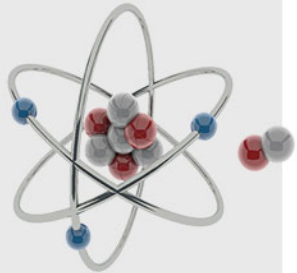
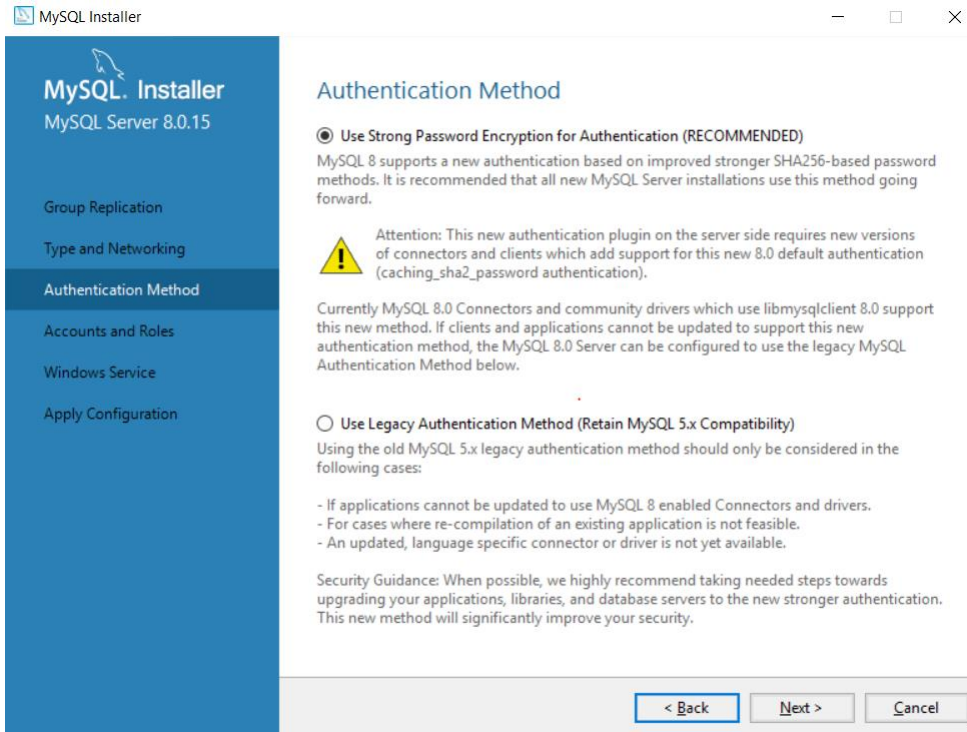
- Server Configuration Type**  
Choose the correct server configuration type for this MySQL Server installation. This setting will define how much system resources are assigned to the MySQL Server instance.  
Config Type: Development Computer
- Connectivity**  
Use the following controls to select how you would like to connect to this server.
  - ☒ TCP/IP Port: 3306 X Protocol Port: 33060
  - ☒ Open Windows Firewall ports for network access
  - ☐ Named Pipe Pipe Name: MYSQL
  - ☐ Shared Memory Memory Name: MYSQL
- Advanced Configuration**  
Select the check box below to get additional configuration pages where you can set advanced and logging options for this server instance.
  - ☐ Show Advanced and Logging Options

At the bottom of the window, there are three buttons: < Back, Next > (highlighted), and Cancel.



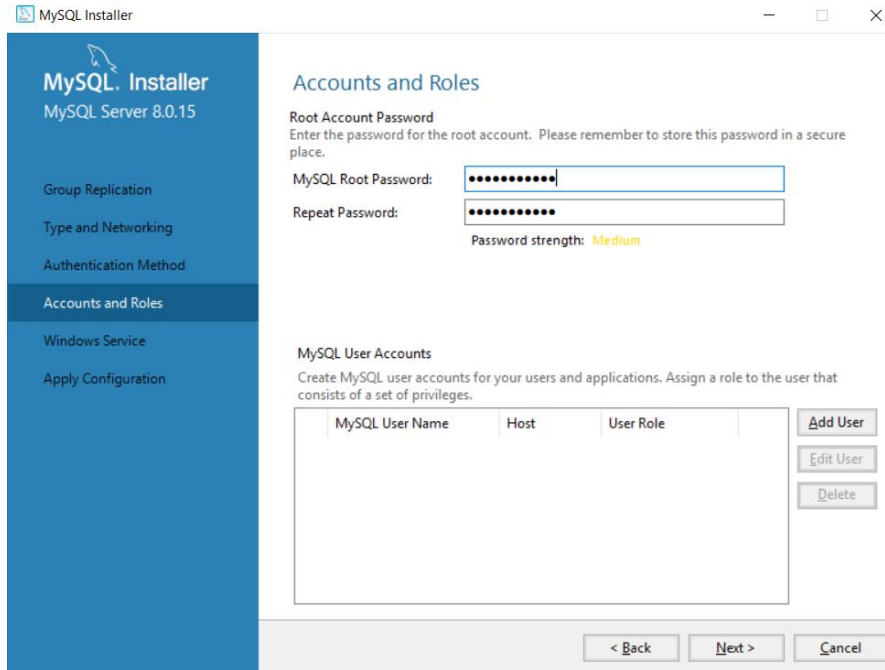
# 3. Hướng dẫn cài đặt MySQL với workbend

## ❑ Bước 6: để mặc định => next



# 3. Hướng dẫn cài đặt MySQL với workbend

## ❑ Bước 7: Xét mặc định cho tài khoản root



MySQL Installer

MySQL Server 8.0.15

- Group Replication
- Type and Networking
- Authentication Method
- Accounts and Roles**
- Windows Service
- Apply Configuration

### Accounts and Roles

**Root Account Password**  
Enter the password for the root account. Please remember to store this password in a secure place.

MySQL Root Password:

Repeat Password:

Password strength: **Medium**

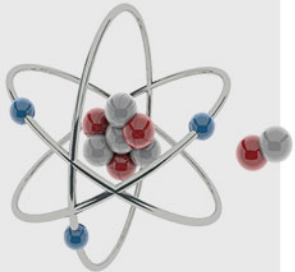
### MySQL User Accounts

Create MySQL user accounts for your users and applications. Assign a role to the user that consists of a set of privileges.

MySQL User Name	Host	User Role
-----------------	------	-----------

[Add User](#)  
[Edit User](#)  
[Delete](#)

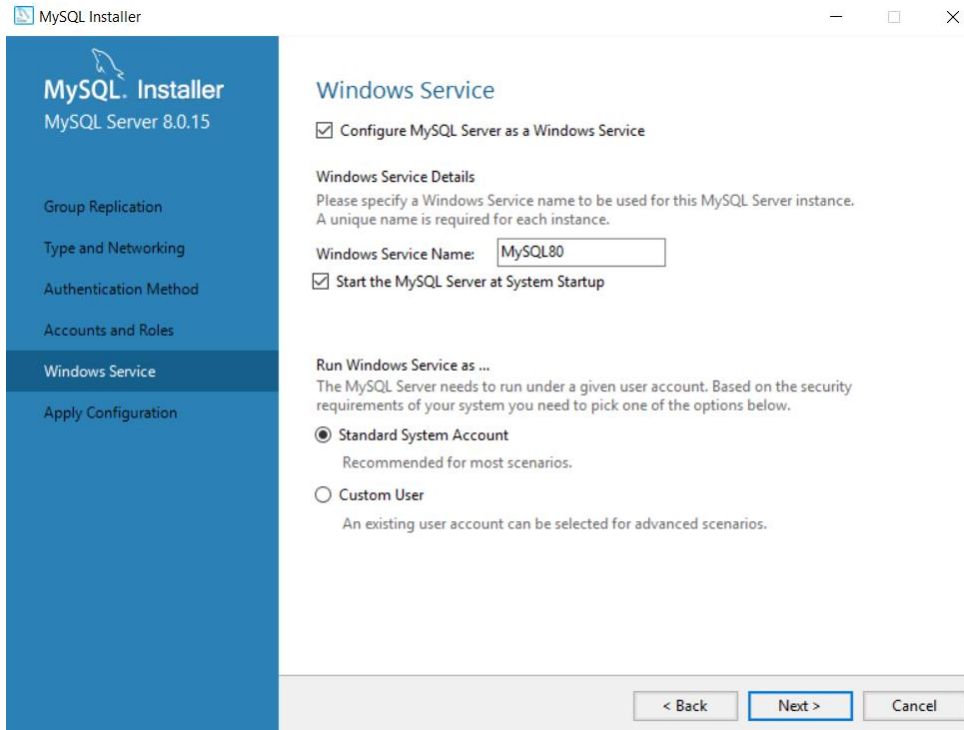
< Back   Next >   Cancel





# 3. Hướng dẫn cài đặt MySQL với workbend

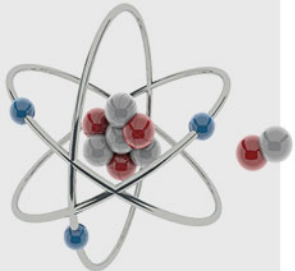
❑ Bước 8: để mặc định như hình => next



The image shows the MySQL Installer window for MySQL Server 8.0.15. The left sidebar contains the following options: Group Replication, Type and Networking, Authentication Method, Accounts and Roles, Windows Service (selected), and Apply Configuration. The main content area is titled 'Windows Service' and includes the following sections:

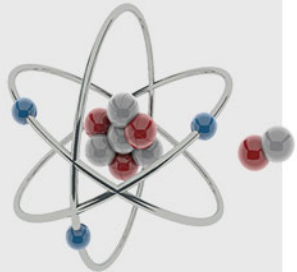
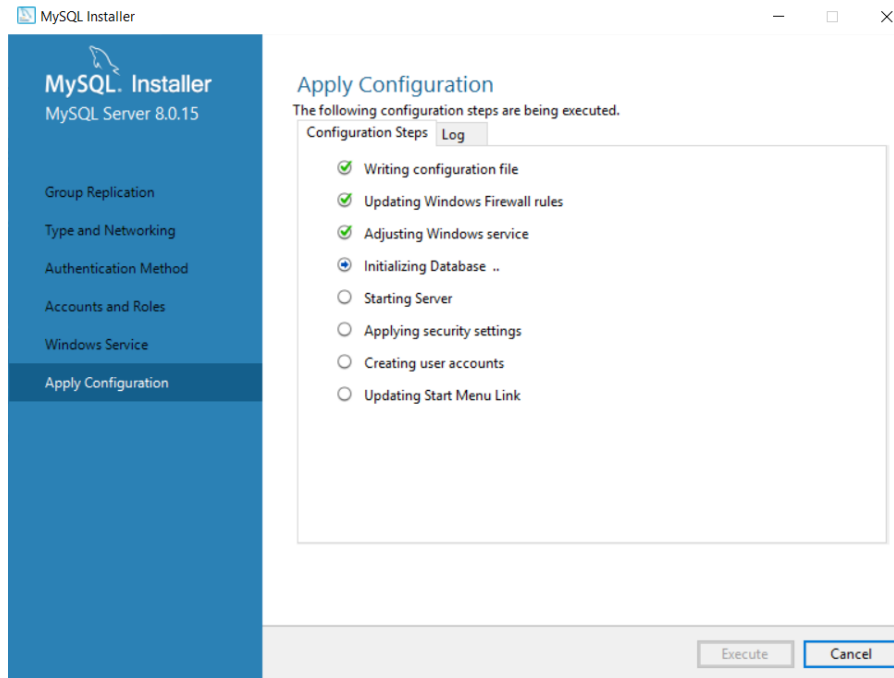
- ☒ Configure MySQL Server as a Windows Service
- Windows Service Details**  
Please specify a Windows Service name to be used for this MySQL Server instance. A unique name is required for each instance.  
Windows Service Name:
- ☒ Start the MySQL Server at System Startup
- Run Windows Service as ...**  
The MySQL Server needs to run under a given user account. Based on the security requirements of your system you need to pick one of the options below.
  - ☒ Standard System Account  
Recommended for most scenarios.
  - ☐ Custom User  
An existing user account can be selected for advanced scenarios.

At the bottom of the window, there are three buttons: '< Back', 'Next >' (highlighted), and 'Cancel'.



# 3. Hướng dẫn cài đặt MySQL với workbend

## ❑ Bước 9: Đợi khoản 5 phút hoàn tất



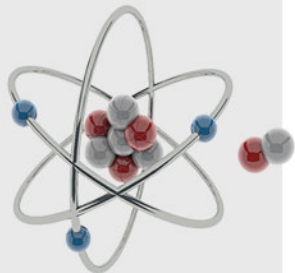
# 4. Ngôn ngữ truy vấn SQL

## SQL là gì ?

- SQL (Structured Query Language) là ngôn ngữ làm việc với CSDL bao gồm nhiều hoạt động như tạo CSDL, thêm mới, cập nhật, xóa, tìm kiếm dữ liệu.
- SQL là ngôn ngữ tiêu chuẩn cho các hệ quản trị CSDL quan hệ.
- Tất cả các hệ thống quản trị CSDL như MySQL, Oracle, Sql Server hay Postgres đều lấy SQL làm ngôn ngữ CSDL tiêu chuẩn.

### ❑ Chức năng.

- Tạo CSDL mới.
- Tạo bảng mới trong CSDL.
- Tạo view (khung nhìn) mới.
- Thực hiện truy vấn trên CSDL.
- Giúp mô tả CSDL.
- Tạo, chèn, xóa, sửa đổi bản ghi trong CSDL.
- Trích xuất dữ liệu từ CSDL.
- Thiết lập quyền trên bảng, thủ tục và view.



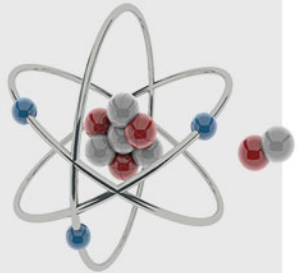
# 4. Ngôn ngữ truy vấn SQL

## ❑ MySQL là gì?

- MySQL là một **cơ sở dữ liệu SQL** mã nguồn mở được phát triển bởi **MySQL AB**, một công ty của Thụy Điển.
- MySQL hỗ trợ nhiều **nền tảng** khác nhau trong đó có các nền tảng như **Window**, **MacOS**, **UNIX** và **Linux**.
- MySQL có cả phiên bản **trả phí** và **miễn phí** tùy thuộc vào mục đích sử dụng (thương mại/phi thương mại).
- MySQL đi kèm với một **máy chủ CSDL SQL** rất nhanh, đa luồng, hỗ trợ nhiều người dùng và cực kỳ mạnh mẽ.

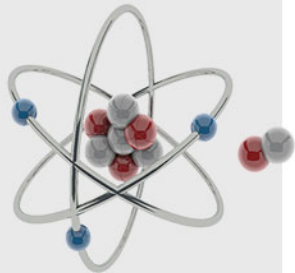
## ❑ Tính năng chính

- **Hiệu năng** cao.
- Tính **khả dụng** cao.
- Khả năng **mở rộng** và **linh hoạt**.
- Kho **dữ liệu web** mạnh.
- Bảo vệ **dữ liệu** mạnh mẽ.
- Phát triển **ứng dụng** toàn diện.
- **Quản lý** dễ dàng.
- **Mã nguồn mở** miễn phí.



## 4. Ngôn ngữ truy vấn SQL

1. Ngôn ngữ định nghĩa dữ liệu (Tạo csdl, tạo bảng, sửa đổi cấu trúc bảng, xóa bảng)
2. Ngôn ngữ thao tác dữ liệu (Thêm, xóa, sửa, truy vấn lồng, gom nhóm và kết hợp)
3. Ngôn ngữ điều khiển dữ liệu (Cấp quyền, thu hồi quyền, từ chối quyền)



# 3.1 Ngôn ngữ định nghĩa dữ liệu



## ❑ Lệnh tạo database (CSDL)

❑ `CREATE DATABASE` [IF NOT EXISTS] `tên_database`;

❖ Trong đó:

✓ `CREATE DATABASE`: Từ khóa tạo database.

✓ `IF NOT EXISTS`: Nếu có dòng này khi tạo bảng trùng tên sẽ không bị báo lỗi.

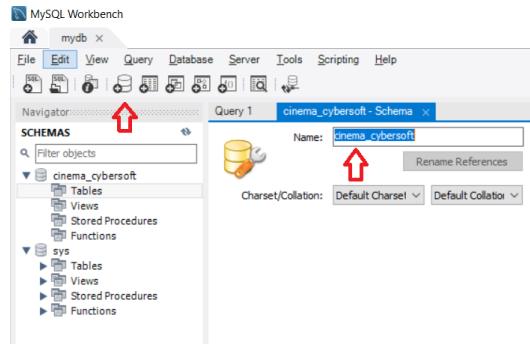
❖ Ví dụ:

```
CREATE DATABASE cinema_cybersoft;
```

Hoặc

```
CREATE DATABASE IF NOT EXISTS cinema_cybersoft;
```

## ❑ Tạo với công cụ



## ❑ Sử dụng database

`USE` `tên_bảng`;

❖ Ví dụ:

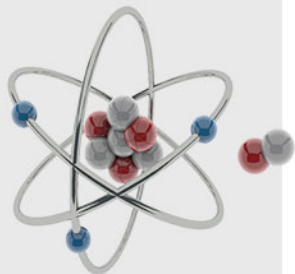
```
USE SINH_VIEN;
```

## ❑ Xóa database

`DROP DATABASE` `tên_bảng`;

❖ Ví dụ:

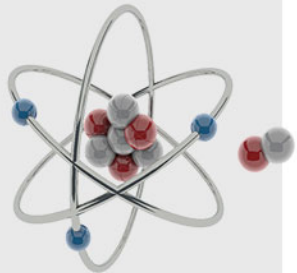
```
DROP DATABASE SINH_VIEN;
```



## 3.1 Ngôn ngữ định nghĩa dữ liệu

### ❑ Quy tắc đặt tên database, bảng, cột.

- Tên **không được** có **số** ở đầu.
- Tên **được phép** có **dấu cách** nhưng phải đặt trong **cặp dấu []**. Ví dụ như [ho ten], [ngay\_sinh], [db sinh vien].
- Tên **được phép** chứa **ký tự có dấu**.
- Tên **được phép** trùng **từ khóa** nhưng phải đặt trong **cặp dấu []**. Ví dụ như [order], [table], [date].
- Tên **không được** chứa dấu **@** hoặc **\$** ở đầu.



## 3.1 Ngôn ngữ định nghĩa dữ liệu

### ❑ Tạo bảng

```
CREATE TABLE [IF NOT EXISTS] tên_bảng (  
    Tên_cột_1 kiểu_dữ_liệu,  
    Tên_cột_2 kiểu_dữ_liệu,  
    ...  
    Tên_cột_n kiểu_dữ_liệu  
);
```

❖ Ví dụ:

```
CREATE TABLE IF NOT EXISTS Phim(  
    MaPhim int,  
    TenPhim nvarchar(50),  
    MoTa nvarchar(255)  
    ...  
);
```

### ❑

### Gán giá trị mặc định

```
CREATE TABLE Phim(  
    MaPhim int,  
    TenPhim nvarchar(50) DEFAULT "Cybersoft",  
    MoTa nvarchar(255)  
);
```

### ❑

### Thiết lập null và not null

```
CREATE TABLE Phim(  
    MaPhim int NOT NULL,  
    TenPhim nvarchar(50),  
    MoTa nvarchar(255)  
);
```

### ❑

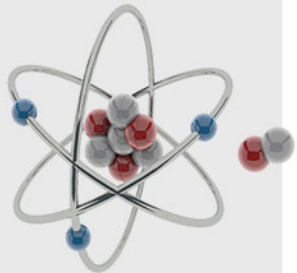
### Xóa bảng

```
DROP TABLE tên_bảng;
```

### ❖

Ví dụ:

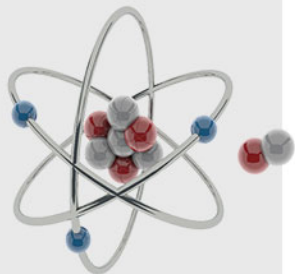
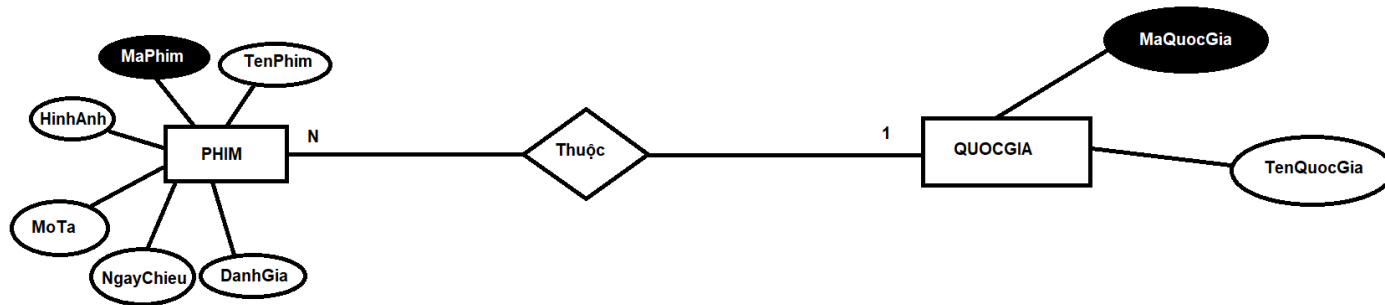
```
DROP TABLE Phim;
```





## 3.1 Ngôn ngữ định nghĩa dữ liệu

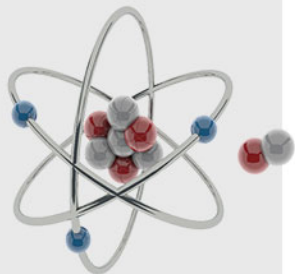
❑ VÍ DỤ: ỨNG VỚI 1 QUAN HỆ TRONG LƯỢT ĐỒ CSDL TƯƠNG DƯƠNG 1 TABLE TRONG CSDL



# 3.1 Ngôn ngữ định nghĩa dữ liệu

## ❑ Tạo bảng bằng lệnh

```
1 CREATE TABLE `cinema_cybersoft`.`phim` (  
2   `MaPhim` INT NOT NULL,  
3   `TenPhim` NVARCHAR(50) NULL,  
4   `HinhAnh` NVARCHAR(50) NULL,  
5   `MoTa` NVARCHAR(50) NULL,  
6   `NgayChieu` DATETIME NULL,  
7   `DanhGia` INT NULL,  
8   PRIMARY KEY (`MaPhim`));  
9
```



# 3.1 Ngôn ngữ định nghĩa dữ liệu

## ❑ Tạo bảng với công cụ

Query 1 new\_table - Table x

Table Name: **Phim** Schema: **cinema\_cybersoft**

Charset/Collation: Default Charset Default Collation Engine: InnoDB

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
MaPhim	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
TenPhim	NVARCHAR(50)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
HinhAnh	NVARCHAR(50)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
MoTa	NVARCHAR(50)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
NgayChieu	DATETIME	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
DanhGia	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Column Name: MaPhim Data Type: INT

Charset/Collation: Default Charset Default Collation

Comments:

Storage: ☐ Virtual ☐ Stored

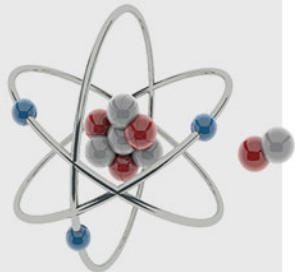
☒ Primary Key ☒ Not Null ☐ Unique

☐ Binary ☐ Unsigned ☐ Zero Fill

☐ Auto Increment ☐ Generated

Columns Indexes Foreign Keys Triggers Partitioning Options

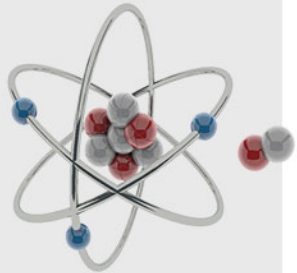
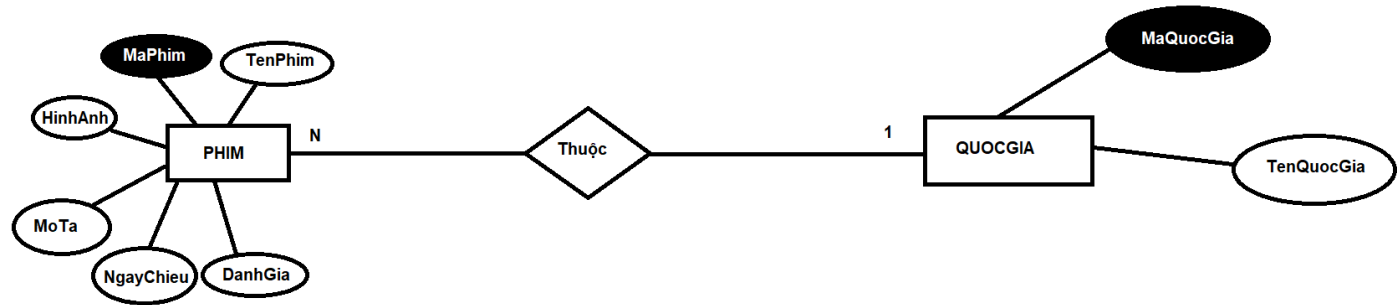
Apply Revert



## 3.1 Ngôn ngữ định nghĩa dữ liệu

❑ Foreign Key là gì? (Định nghĩa khóa ngoại)

- **Khóa ngoại** (**Foreign Key**) được dùng để **kết nối** hai bảng lại với nhau.
- **Khóa ngoại** là một trường hoặc nhiều trường trong 1 bảng tham chiếu tới **khóa chính** của bảng khác.
- Bảng chứa **khóa ngoại** được gọi là **bảng con** và bảng chứa **khóa ứng viên** được gọi là **bảng được tham chiếu** hoặc gọi là **bảng cha**.
- Lấy ví dụ trên: 1 phim được sản xuất (thuộc) bởi 1 quốc gia. 1 quốc gia có 1 quốc gia có thể có nhiều phim.



# 3.1 Ngôn ngữ định nghĩa dữ liệu



## ❑ Cú pháp tạo khóa ngoại tham chiếu

- ❑ Sử dụng CONSTRAINT (Trực tiếp khi tạo bảng)

```
CREATE TABLE Phim (  
    MaPhim int NOT NULL PRIMARY KEY,  
    TenPhim nvarchar(255) NOT NULL,  
    ...  
    MaQuocGia int NOT NULL,  
    CONSTRAINT [ten_rang_buoc] FOREIGN KEY (MaQuocGia) REFERENCES QuocGia (MaQuocGia)  
);
```

- ❑ Xoá khóa ngoại

```
ALTER TABLE Products DROP FOREIGN KEY fk_pro;
```

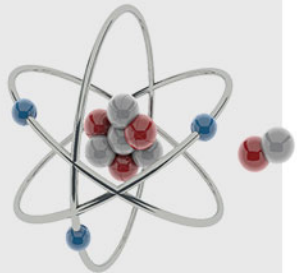
- ❑ Sử dụng ALTER TABLE (sau khi lỡ tạo bảng rồi)

```
CREATE TABLE Phim (  
    MaPhim int NOT NULL PRIMARY KEY,  
    TenPhim nvarchar(255) NOT NULL,  
    ...  
    MaQuocGia int NOT NULL);
```

```
ALTER TABLE Phim ADD FOREIGN KEY (MaQuocGia)  
REFERENCES QuocGia (MaQuocgia);
```

Hoặc

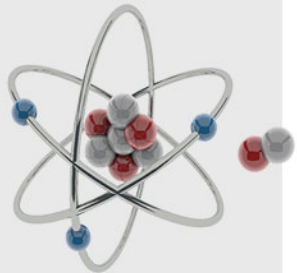
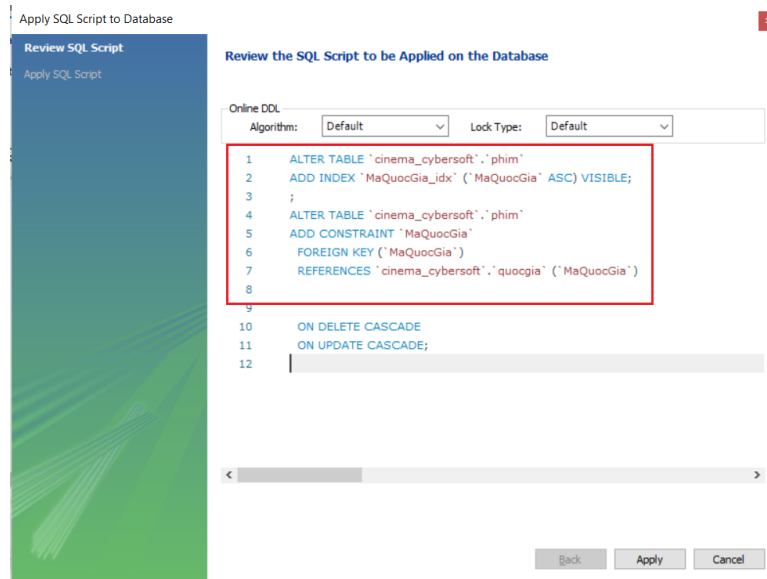
```
ALTER TABLE ADD CONSTRAINT ten_rang_buoc  
FOREIGN KEY (MaQuocGia) REFERENCES QuocGi(MaQuocGia);
```



## 3.1 Ngôn ngữ định nghĩa dữ liệu

❑ VÍ DỤ: Chuyển đổi quan hệ 1 – n sang cơ sở dữ liệu theo quy tắc khóa chính của bảng 1 → khóa ngoại của bảng nhiều.

❑ Lưu ý: **Tạo theo thứ tự bảng chính tạo trước, bảng chứa khóa ngoại tham chiếu tạo sau. Kiểu dữ liệu của trường tham chiếu phải là kiểu dữ liệu khóa chính của bảng chính.**



# 3.1 Ngôn ngữ định nghĩa dữ liệu

## ❑ VÍ DỤ: Thực hiện bằng công cụ

phim - Table x quocgia - Table

Table Name:  Schema: **cinema\_cybersoft**

Charset/Collation:   Engine:

Comments:

Foreign Key Name	Referenced Table	Column	Referenced Column
MaQuocGia	"cinema_cybersoft"."quocgia"	<input type="checkbox"/> MaPhim <input type="checkbox"/> TenPhim <input type="checkbox"/> HinhAnh <input type="checkbox"/> MoTa <input type="checkbox"/> NgayChieu <input type="checkbox"/> DanhGia <input checked="" type="checkbox"/> MaQuocGia	MaQuocGia

Foreign Key Options

On Update:

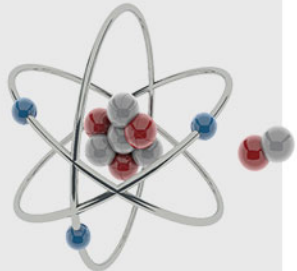
On Delete:

☐ Skip in SQL generation

Foreign Key Comment:

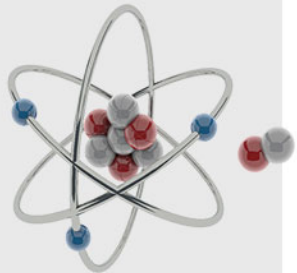
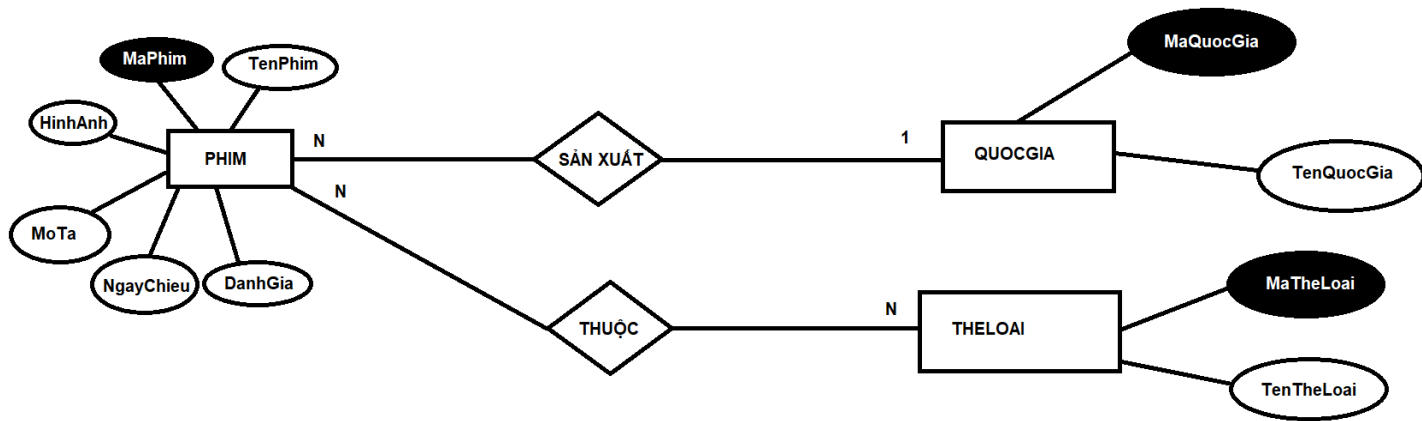
Columns Indexes **Foreign Keys** Triggers Partitioning Options

Apply Revert



## 3.1 Ngôn ngữ định nghĩa dữ liệu

❑ VÍ DỤ: Chuyển đổi quan hệ n – n sang cơ sở dữ liệu theo quy tắc sinh ra quan hệ (bảng mới). Table mới sẽ lấy 2 khóa ngoại tham chiếu từ 2 bảng chính làm khóa chính cho bảng đó. (Sử dụng cặp khóa chính).





## 3.1 Ngôn ngữ định nghĩa dữ liệu

❑ **VÍ DỤ:** Ta có thêm bảng TheLoai(MaTheLoai, TenTheLoai). Ta có quan hệ thứ 3 sinh ra để lưu trữ thông tin PhimTheLoai(MaPhim, MaTheLoai) Cặp khóa chính này tham chiếu từ 2 bảng

❑ Chọn nhiều khóa chính

```
CREATE TABLE Phim_TheLoai(  
    MaPhim int NOT NULL,  
    MaTheLoai nvarchar(50),  
    PRIMARY KEY (MaPhim, MaTheLoai)  
);
```

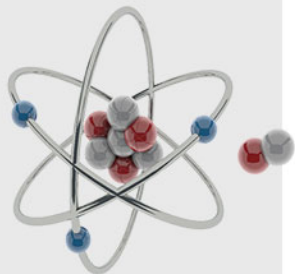
❑ Sử dụng ALTER

```
CREATE TABLE Phim_TheLoai(  
    MaPhim int NOT NULL,  
    MaTheLoai nvarchar(50),  
    );  
  
ALTER TABLE Phim_TheLoai ADD PRIMARY KEY (MaPhim, MaTheLoai);
```

❑ Sử dụng ALTER add tham chiếu

```
ALTER TABLE `cinema_cybersoft`.`phim_theloai` ADD CONSTRAINT `MaPhim` FOREIGN KEY (`MaPhim` , `MaPhim`)  
REFERENCES `cinema_cybersoft`.`phim` (`MaPhim` , `MaPhim`)
```

```
ALTER TABLE `cinema_cybersoft`.`phim_theloai` ADD CONSTRAINT `MaTheLoai` FOREIGN KEY (`MatheLoai` , `MatheLoai`)  
REFERENCES `cinema_cybersoft`.`TheLoai` (`MaTheLoai` , `MaTheLoai`)
```



# BÀI TẬP

## ❑ Bài tập 1:

- Tạo bảng Students gồm các cột Id, Full\_name, Gender, Age, City, Weight.

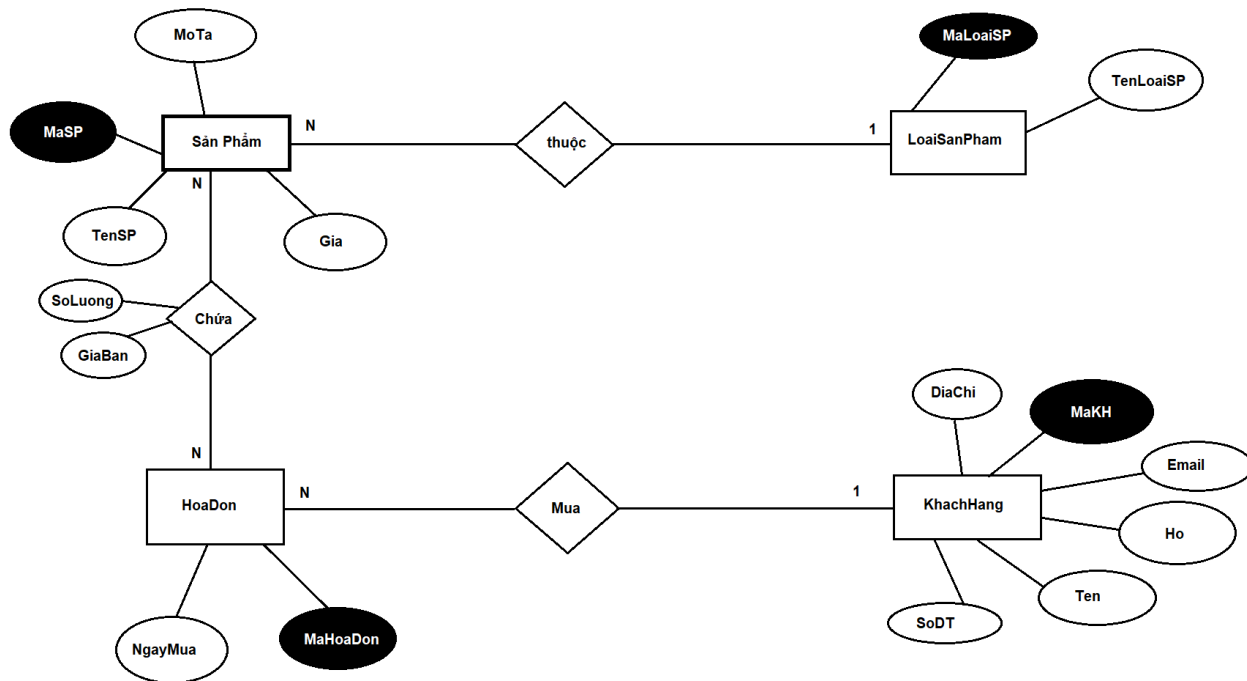
Trong đó:

- Id có kiểu int, tự động tăng và là khóa chính của bảng.
- Full\_name mang kiểu chuỗi có dấu và không được null.
- Gender mang kiểu chuỗi không có dấu.
- Age có kiểu int
- City có mang kiểu chuỗi có dấu.
- Weight mang kiểu số thập phân.

Full_name	Gender	Age	City	Weight
Nguyen Thanh Nhan	Nam	19	Can Tho	56.5674
Pham Thu Huong	Nu	20	Vinh Long	72.456
Nguyen Nhu Ngoc	Nu	20	Soc Trang	85.387
Bui Thanh Bao	Nam	19	Soc Trang	49.3
Le My Nhan	Nu	22	Can Tho	62.963
Tan Thuc Bao	Nam	35	An Giang	55.5678
Trinh Giao Kim	Nam	44	Bac Lieu	67.34

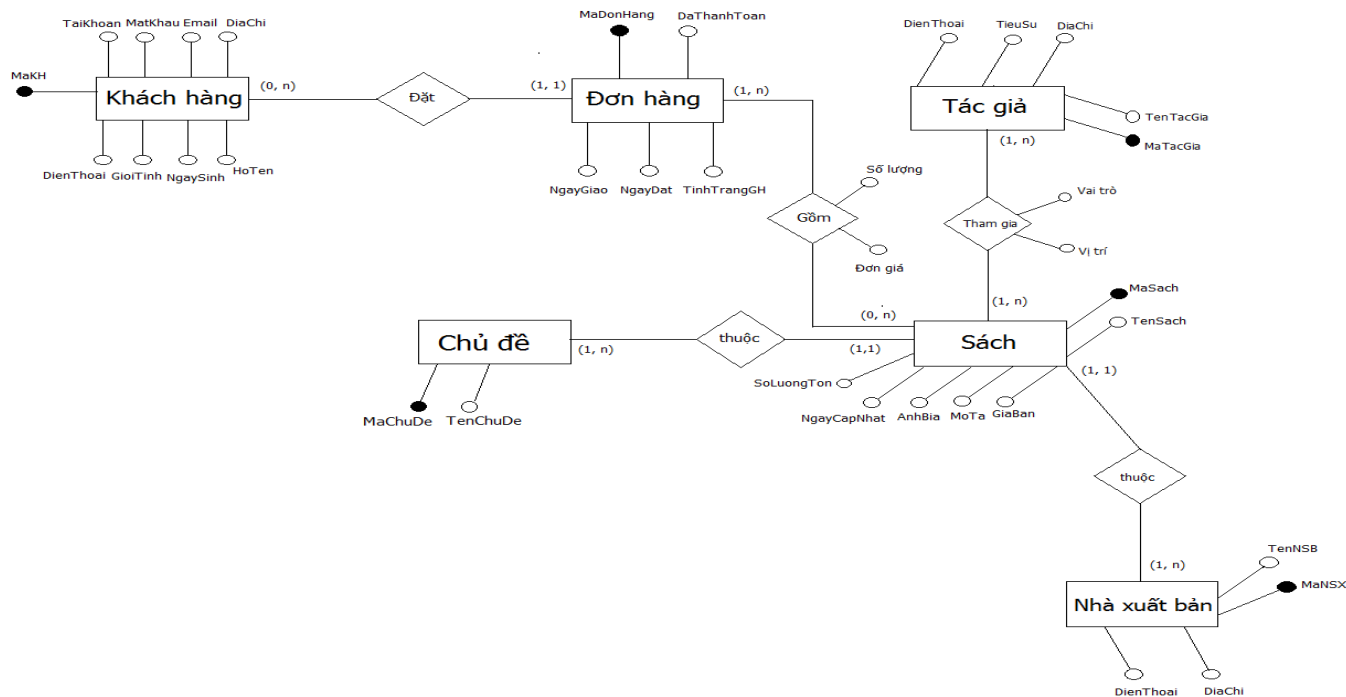
# BÀI TẬP

- Bài tập 2: Dựa vào lược đồ ERD hình bên dưới hãy thiết kế cơ sở dữ liệu tương ứng đặt tên db là cybersoft\_shop



# BÀI TẬP VỀ NHÀ

- Bài tập 2: Dựa vào lược đồ ERD hình bên dưới hãy thiết kế cơ sở dữ liệu tương ứng đặt tên db QLBanSach



## 3.2 Ngôn ngữ thao tác dữ liệu



### ❏ Câu lệnh INSERT

➤ Dùng để thêm mới một hàng dữ liệu mới vào bảng trong CSDL.

❖ Cú pháp:

```
INSERT INTO tên_bảng (cột_1, cột_2, ..., cột_n)
VALUES (giá_trị_1, giá_trị_2, ..., giá_trị_n);
```

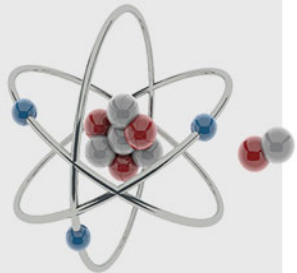
❖ Ví dụ:

✓ Tạo bảng:

```
CREATE TABLE Users(
    Id varchar(50) NOT NULL PRIMARY KEY,
    Email varchar(100) NOT NULL UNIQUE,
    FullName nvarchar(255)
);
```

✓ Thêm dữ liệu:

```
INSERT INTO Users (Id, Email, FullName) VALUES ("nv001", "nv001@gmail.com", "Nguyễn Văn A");
INSERT INTO Users (Id, Email, FullName) VALUES ("nv002", "nv002@gmail.com", "Trần Văn B");
```



## 3.2 Ngôn ngữ thao tác dữ liệu

### ❑ Câu lệnh SELECT

➤ Câu lệnh **SELECT** dùng để lấy dữ liệu của một hoặc nhiều bảng trong CSDL.

❖Cú pháp:

```
SELECT field1, field2, field3, ...  
FROM table_name  
WHERE <dieu_kien_loc>  
ORDER BY field_name ASC|DESC  
LIMIT start, limit
```

### ❑ Câu lệnh SELECT

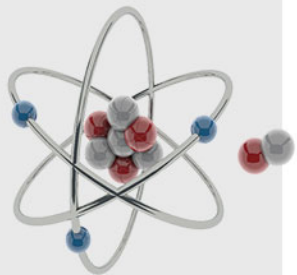
➤ **SELECT** field1, field2, field3, ... là danh sách các fields cần lấy

➤ **FROM** table\_name tên table cần lấy

➤ **WHERE** <dieu\_kien\_loc> là các điều kiện để lấy các dòng dữ liệu

➤ **ORDER BY** field\_name, ASC|DESC: là cách sắp xếp cho field\_name theo kiểu ASC (tăng dần) hoặc DESC (giảm dần)

➤ **LIMIT** start, limit là lấy limit records kể từ record thứ start trong kết quả.



## 3.2 Ngôn ngữ thao tác dữ liệu

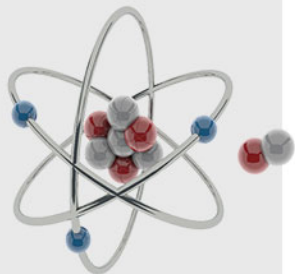
❑ Ví dụ

❖ Tạo bảng:

❖ Truy vấn:

```
SELECT sv_id, sv_name, sv_description  
FROM SINHVIEN  
WHERE sv_id < 4  
ORDER BY sv_id DESC  
LIMIT 0, 2;
```

	sv_id	sv_name	sv_description
<input type="checkbox"/>	1	Mr Cuong	Nguyen Van Cuong
<input type="checkbox"/>	2	Mr Kinh	Nguyen Van Kinh
<input type="checkbox"/>	3	Mr Chinh	Nguyen Van Chinh
<input type="checkbox"/>	4	Mr Quyen	Nguyen Van Quyen



# WHERE

## ❑ Mệnh đề WHERE

- Mệnh đề **WHERE** được sử dụng để **lọc** các **bản ghi** và **chỉ lấy** những **bản ghi phù hợp** với yêu cầu hoặc thực sự **cần thiết**.
- Mệnh đề **WHERE** không đi một mình mà thường **kết hợp** với câu lệnh **SELECT**, **UPDATE**, **DELETE** hoặc các câu lệnh khác.

## ❑ Cú pháp

```
SELECT cot1, cot2, cotN  
FROM tên_bảng  
WHERE [điều_kiện];
```

- **[điều\_kiện]** : Có thể sử dụng các toán tử logic hoặc so sánh như <, >, LIKE, NOT, IN, ...



# ORDER BY

## ❑ Lệnh ORDER BY

- Lệnh **ORDER BY** dùng chung với lệnh **SELECT** để **sắp xếp kết quả trả về** theo tiêu chí.

### ❖ Cú pháp:

```
SELECT cột1, cột2...  
FROM tên_bảng  
WHERE [điều_kiện]  
ORDER BY tên_cột kiểu_sắp_xếp
```

### ❖ Chú thích:

- ✓ **tên\_cột**: tên column cần sắp xếp.
- ✓ **kiểu\_sắp\_xếp**: loại sắp xếp và có giá trị là:  
**ASC** nếu tăng dần.  
**DESC** nếu giảm dần.

### ❖ Ví dụ:

- Sắp xếp tăng dần  

```
SELECT MaSV, TenSV, NamSinh  
FROM SINHVIEN  
ORDER BY MaSV ASC.
```
- Sắp xếp giảm dần  

```
SELECT MaSV, TenSV, NamSinh  
FROM SINHVIEN  
ORDER BY MaSV DESC.
```

## ❑ Lệnh LIMIT

- Lệnh **LIMIT** đi kèm với lệnh **SELECT** và thông thường nó **nằm ở cuối cùng**.
- Trong thuật toán **phân trang** sẽ sử dụng lệnh **LIMIT** để xác định **kết quả** cho mỗi **trang**.
- Lệnh **LIMIT** giúp **tăng tốc độ load** trang hơn.

## ❖ Cú pháp:

```
SELECT cột1, cột2, cột3, ...  
FROM tên_bảng  
WHERE <điều_kiện>  
ORDER BY tên_cột ASC|DESC  
LIMIT bắt_đầu, số_lượng;
```

➤ **ORDER BY:** Kiểu sắp xếp.

➤ **LIMIT:**

**bắt\_đầu:** lấy từ bản ghi thứ **bắt\_đầu**.

**số\_lượng:** bắt đầu lấy từ bản ghi thứ **bắt\_đầu** và lấy tiếp **số\_lượng** bản ghi.

## ❑ Toán tử AND

- Toán tử **AND** cho phép **sử dụng** nhiều **điều kiện** trong mệnh đề **WHERE** của câu lệnh **SQL**.

### ❖ Cú pháp:

```
SELECT cot1, cot2, cotN
FROM tên_bảng
WHERE [điều_kiện1] AND [điều_kiện2]... A
ND [điều_kiệnN];
```

### ❖ Ví dụ:

```
SELECT Id, Name, Age, Salary
FROM Employees
WHERE Age > 18 AND age < 30;
```

## ❑ Toán tử OR

- Toán tử **OR** được sử dụng để **kết hợp** nhiều **điều kiện** trong mệnh đề **WHERE** của lệnh **SQL**.

### ❖ Cú pháp:

```
SELECT cot1, cot2, cotN
FROM tên_bảng
WHERE [điều_kiện1] OR [điều_kiện2]...
OR [điều_kiệnN];
```

### ❖ Ví dụ:

```
SELECT Id, Name, Age, Salary
FROM Employees
WHERE Salary > 2000 OR age < 3
```

0;

## ❑ Bài tập 2:

- Sử dụng bảng vừa tạo ở bài tập 1 và thêm dữ liệu vào bảng như hình bên.
- Truy vấn và hiển thị thông tin các sinh viên có giới tính là nam, sắp xếp theo thứ tự tuổi giảm dần.
- Truy vấn và hiển thị thông tin hai sinh viên đầu tiên có giới tính là nữ.
- Truy vấn và hiển thị thông tin họ tên, tuổi các sinh viên ở Cần Thơ hoặc Sóc trăng.

Full_name	Gender	Age	City	Weight
Nguyen Thanh Nhan	Nam	19	Can Tho	56.5674
Pham Thu Huong	Nu	20	Vinh Long	72.456
Nguyen Nhu Ngoc	Nu	20	Soc Trang	85.387
Bui Thanh Bao	Nam	19	Soc Trang	49.3
Le My Nhan	Nu	22	Can Tho	62.963
Tan Thuc Bao	Nam	35	An Giang	55.5678
Trinh Giao Kim	Nam	44	Bac Lieu	67.34

## ❑ Bảng dữ liệu mẫu

➤ Bảng Students:

Full_name	Gender	Age	City	Weight
Nguyen Thanh Nhan	Nam	19	Can Tho	56.5674
Pham Thu Huong	Nu	20	Vinh Long	72.456
Nguyen Nhu Ngoc	Nu	20	Soc Trang	85.387
Bui Thanh Bao	Nam	19	Soc Trang	49.3
Le My Nhan	Nu	22	Can Tho	62.963
Tan Thuc Bao	Nam	35	An Giang	55.5678
Trinh Giao Kim	Nam	44	Bac Lieu	67.34

# SQL FUNCTION

## ❑ Hàm AVG()

➤ Hàm **AVG()** dùng để lấy **giá trị trung bình cộng** của một cột.

❖ Cú pháp:

```
SELECT AVG(tên_cột)
FROM tên_bảng;
```

❖ Ví dụ:

✓ Lấy tuổi trung bình của các sinh viên trong bảng:

```
SELECT AVG(Age)
FROM Students;
```

## ❑ Hàm COUNT()

➤ Hàm **COUNT()** được dùng để **đếm số lượng mẫu tin** (dữ liệu, hàng) trong **bảng**.

❖ Cú pháp:

```
SELECT COUNT(*)
FROM tên_bảng;
```

❖ Ví dụ:

✓ Đếm số lượng sinh viên có giới tính là nam :

```
SELECT COUNT(*)
FROM Students
WHERE Gender = "Nam";
```

# SQL FUNCTION

## ❑ Hàm MIN() và MAX()

➤ Hàm **MAX()** dùng để lấy **giá trị lớn nhất** trong một cột.

➤ Hàm **MIN()** dùng để lấy **giá trị nhỏ nhất** trong một cột.

✓ Lưu ý: Giá trị có thể là kiểu số hoặc kiểu chuỗi, ký tự,....

❖ Cú pháp:

```
SELECT MAX(tên_cột)
FROM tên_bảng;
```

Lấy min:

```
SELECT MIN(tên_cột)
FROM tên_bảng;
```

❖ Ví dụ:

✓ Lấy sinh viên nữ lớn tuổi nhất

```
SELECT MAX(Age)
FROM Students
WHERE Gender = 'Nu';
```

✓ Lấy sinh viên nam nhỏ tuổi nhất

```
SELECT MIN(Age)
FROM Students
WHERE Gender = 'Nam';
```

# SQL FUNCTION

## ❑ Hàm SUM()

➤ Hàm **SUM()** dùng để lấy **giá trị tổng** của một cột.

❖ Cú pháp:

```
SELECT SUM(tên_cột)
FROM tên_bảng;
```

❖ Ví dụ:

✓ Lấy giá trị tổng của cột tuổi sinh viên.

```
SELECT SUM(Age)
FROM Students;
```

## ❑ Hàm ROUND()

➤ Hàm **ROUND()** được dùng để **làm tròn số thập phân**.

❖ Cú pháp:

```
SELECT
ROUND(tên_cột, số_lượng_thập_p
hân)
```

```
FROM tên_bảng;
```

❖ Ví dụ:

✓ Lấy cột họ tên và cân nặng của sinh viên, làm tròn cân nặng đến số thập phân thứ 2.

```
SELECT Full_name,
ROUND(Weight,2)
FROM Students;
```



# GROUP BY

## ❑ Mệnh đề GROUP BY

- Mệnh đề **GROUP BY** được sử dụng kết hợp với lệnh **SELECT** để **sắp xếp dữ liệu từ nhiều bản ghi** đồng nhất vào trong các **nhóm**.
- Mệnh đề **GROUP BY** đứng sau mệnh đề **WHERE** trong câu lệnh **SELECT** và đứng trước mệnh đề **ORDER BY**.

## ❖ Cú pháp:

```
SELECT cot1, cot2  
FROM ten_bang  
WHERE [ dieu_kien ]  
GROUP BY cot1, cot2  
ORDER BY cot1, cot2
```

# GROUP BY

## ❑ Ví dụ

➤ Bảng NHANVIEN:

ID	TEN	TUOI	DIACHI	LUONG
1	Thanh	32	Haiphong	2000.00
2	Thanh	25	Hanoi	1500.00
3	Nga	23	Hanam	2000.00
4	Nga	25	Hue	6500.00
5	Huy	27	Hatinh	8500.00
6	Cao	22	HCM	4500.00
7	Lam	24	Hanoi	10000.00

- ✓ Lấy ra tổng số tiền lương của mỗi nhân viên.

```
SELECT TEN, SUM(LUONG)
FROM NHANVIEN
GROUP BY TEN;
```

➤ Kết quả:

TEN	SUM(LUONG)
Huy	8500.00
Nga	8500.00
Cao	4500.00
Lam	10000.00
Thanh	3500.00

## ❑ Mệnh đề HAVING

- Mệnh đề **HAVING** trong **SQL** được sử dụng để **lọc các bản ghi** và chỉ lấy những **bản ghi phù hợp với yêu cầu** hoặc thực sự **cần thiết tương tự** như mệnh đề **WHERE**. Tuy nhiên:
- Mệnh đề **WHERE** là câu lệnh điều kiện trả **kết quả đối chiếu** với từng **dòng**.
- Mệnh đề **HAVING** là câu lệnh điều kiện trả kết **quả đối chiếu** cho **nhóm** được tạo bởi mệnh đề **GROUP BY**.

❖ Cú pháp:

```
SELECT cot1, cot2  
FROM bang1, bang2  
WHERE [ dieu_kien ]  
GROUP BY cot1, cot2  
HAVING [ dieu_kien ]  
ORDER BY cot1, cot2
```

# HAVING

## ❑ Ví dụ

ID	TEN	TUOI	DIACHI	LUONG
1	Thanh	32	Haiphong	2000.00
2	Loan	25	Hanoi	1500.00
3	Nga	23	Hanam	2000.00
4	Manh	25	Hue	6500.00
5	Huy	27	Hatinh	8500.00
6	Cao	22	HCM	4500.00
7	Lam	24	Hanoi	10000.00

- ✓ Hiển thị bản ghi có độ tuổi xuất hiện từ 2 lần trở lên.

```
SELECT ID, TEN, TUOI, DIACHI, LUONG
FROM NHANVIEN
GROUP BY tuoi
HAVING COUNT(tuoi) >= 2;
```

➤ Kết quả:

ID	TEN	TUOI	DIACHI	LUONG
2	Loan	25	Hanoi	1500.00

# UPDATE

## ❑ Câu lệnh UPDATE

- Dùng để **cập nhật** một **dòng dữ liệu** trong **bảng** của **cơ sở dữ liệu**.

### ❖ Cú pháp

```
UPDATE tên_bảng  
SET cột_1 = giá_trị_1, ..., cột_n = giá_trị_n  
WHERE điều_kiện;
```

## ❑ Ví dụ:

- ✓ Tạo bảng:

```
CREATE TABLE Users(  
    Id varchar(50) NOT NULL PRIMARY  
KEY,  
    Email varchar(100) NOT NULL UNIQUE,  
    FullName nvarchar(255)  
);
```

- ✓ Thêm mới:

```
INSERT INTO Users (Id, Email, FullName) VALUES ("nv001", "nv001@gmail.com", "Nguyễn Văn A");
```

- ✓ Cập nhật:

```
UPDATE Users SET Email = "tranteo@gmail.com", FullName = "Trần Tèo" WHERE Id = "nv001";
```

# DELETE

## ❑ Câu lệnh DELETE

- Lệnh **DELETE** được sử dụng để **xóa** những **bản ghi** đang tồn tại trong một **bảng**.
- Sử dụng mệnh đề **WHERE** với lệnh **DELETE** để **xóa** với **điều kiện**, nếu không, **tất cả** các **bản ghi** sẽ bị xóa.

## ❖ Cú pháp:

```
DELETE FROM tên_bảng  
WHERE [điều_kiện];
```

## ❖ Ví dụ:

- Xóa hết bản ghi trong bảng  
**DELETE FROM** Customers;
- Xóa với điều kiện  
**DELETE FROM** Customers  
**WHERE** Id = 3;

# BÀI TẬP

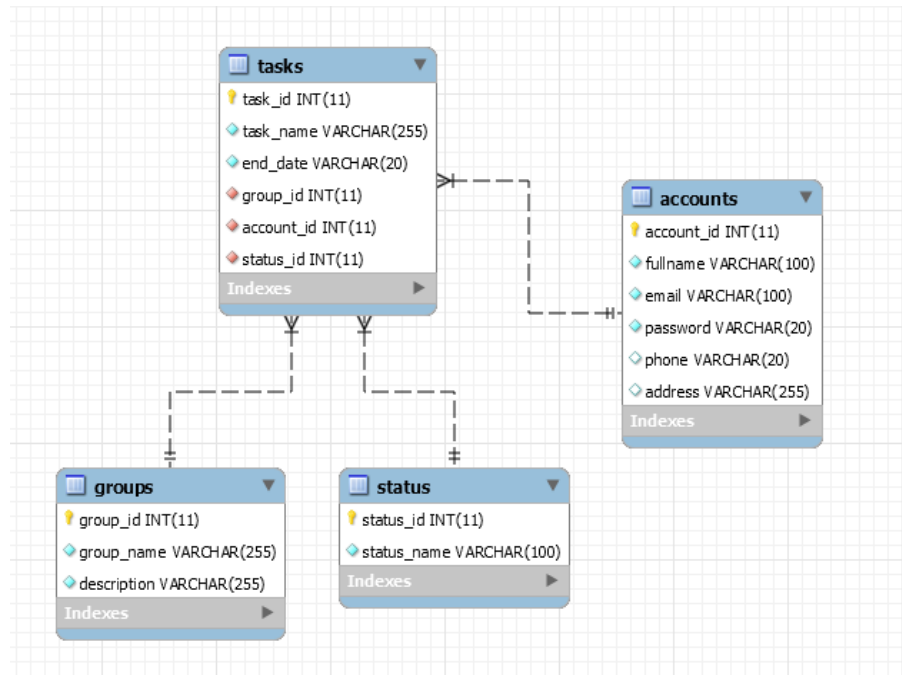
## ❑ Bài tập CRM

1. Tạo CSDL có tên CRM\_DB với các bảng như hình bên.

- Bảng tasks và groups có quan hệ 1 – n.
- Bảng tasks và accounts có quan hệ 1 – n.
- Bảng tasks và status có quan hệ 1 – n.

2. Thêm mới dữ liệu mẫu vào các bảng.

3. Thực hiện một số câu truy vấn.



# Kết nối các bảng với các mệnh đề



## ❑ Nội dung.

- Mệnh đề JOIN là gì.
- Mệnh đề INNER JOIN.
- Mệnh đề LEFT JOIN.
- Mệnh đề RIGHT JOIN.
- Mệnh đề FULL JOIN.
- Câu lệnh SUBQUERY.
- Sử dụng SUBQUERY với SELECT.
- Sử dụng SUBQUERY với INSERT.
- Sử dụng SUBQUERY với UPDATE.
- Sử dụng SUBQUERY với DELETE.
- Thực hành các câu lệnh trong dự án Elearning.
- Thực hành các câu lệnh trong dự án Quản lý phim.



## ❑ JOIN là gì?

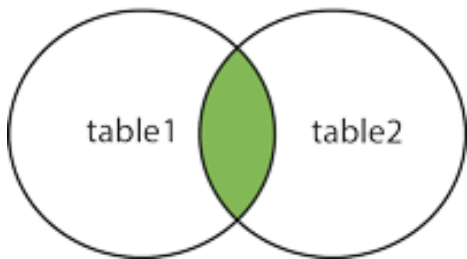
- Mệnh đề **JOIN** được sử dụng để **kết hợp** các **bản ghi** từ **hai hoặc n** **hiệu bảng** trong một **Database** bằng cách **sử dụng** các **giá trị chủ** từ mỗi **bảng**.
- Mệnh đề **JOIN** được thực hiện trong mệnh đề **WHERE**. Một số **toán tử** có thể được sử dụng để **kết hợp** các **bảng** là: =, <, >, <>, <=, >=, !=, BETWEEN, LIKE, và NOT.
- **Toán tử** được sử dụng **phổ biến nhất** là dấu bằng (=).
- Sử dụng SUBQUERY với UPDATE.
- Sử dụng SUBQUERY với DELETE.
- Thực hành các câu lệnh trong dự án Elearning.
- Thực hành các câu lệnh trong dự án Quản lý phim.

# INNER JOIN

## ❑ Mệnh đề INNER JOIN

- Mệnh đề **INNER JOIN** trong **SQL** là kiểu **JOIN** quan trọng và thường được sử dụng nhiều nhất.
- Mệnh đề **INNER JOIN** truy vấn với kết quả trả về là **tập hợp các dữ liệu** thỏa **mãn điều kiện chung** từ hai bảng.

INNER JOIN



## ❖ Cú pháp:

```
SELECT cot1, cot2,... cotn
```

```
FROM bang1
```

```
INNER JOIN bang2
```

```
ON bang1.cot_chung = bang2.co  
t_chung;
```

## ➤ Trong đó:

- ✓ **cot1, cot2,... cotn**: tên các **cột** cần hiển thị ở kết quả truy vấn. Các cot được ngăn cách với nhau bằng dấu phẩy (,)
- ✓ **bang1, bang2**: tên các **bảng** để lấy dữ liệu khi truy vấn.
- ✓ **cot\_chung**: thường là **tên cột khóa ngoại** tham chiếu từ bang1 đến cột định danh trong bang2 hoặc ngược lại.

# INNER JOIN

## ❑ Ví dụ

### ❖ Bảng nhân viên

ID	TEN	TUOI	DIACHI	LUONG
1	Thanh	32	Haiphong	2000.00
2	Loan	25	Hanoi	1500.00
3	Nga	23	Hanam	2000.00
4	Manh	25	Hue	6500.00

### ❖ Bảng tiền thưởng

TT_ID	NGAY	NHANVIEN_ID	SOTIEN
102	2019-01-08 00:00:00	3	3000
100	2019-01-08 00:00:00	3	1500
101	2019-02-20 00:00:00	2	1560
103	2018-12-20 00:00:00	4	2060

➤ JOIN hai bảng bằng cách sử dụng INNER JOIN như sau:

```
SELECT ID, TEN, SOTIEN, NGAY  
FROM NHANVIEN  
INNER JOIN TIENTHUONG  
ON NHANVIEN.ID =  
TIENTHUONG.NHANVIEN_
```

ID;

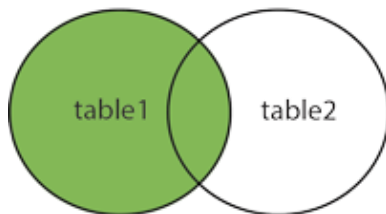
ID	TEN	SOTIEN	NGAY
3	Nga	3000	2019-01-08 00:00:00
3	Nga	1500	2019-01-08 00:00:00
2	Loan	1560	2019-02-20 00:00:00
4	Manh	2060	2018-12-20 00:00:00

# LEFT JOIN

## ❑ Mệnh đề LEFT JOIN

- Mệnh đề **LEFT JOIN** trong **SQL** là kiểu **JOIN** trả về **tất cả** các **bản ghi** từ **bảng bên trái** (bảng 1) và các **bản ghi** phù hợp từ **bảng bên phải** (bảng 2).
- Nếu mệnh đề **ON** không khớp với **bản ghi** nào trong **bảng** bên phải thì **LEFT JOIN** sẽ vẫn trả về **một hàng** trong kết quả, nhưng giá trị là **NULL** trong mỗi cột từ **bảng bên phải**.

LEFT JOIN



## ❖ Cú pháp:

```
SELECT cot1, cot2,... cotn  
FROM bang1  
LEFT JOIN bang2  
ON bang1.cot_chung = bang2.cot_chung;
```

## ➤ Trong đó:

- ✓ **cot1, cot2,... cotn**: tên các cột cần hiển thị ở kết quả truy vấn. Các cot được ngăn cách với nhau bằng dấu phẩy (,)
- ✓ **bang1, bang2**: tên các bảng để lấy dữ liệu khi truy vấn.
- ✓ **cot\_chung**: thường là tên cột khóa ngoại tham chiếu từ bang1 đến cột định danh trong bang2 hoặc ngược lại.

# LEFT JOIN

## ❏ Ví dụ

ID	TEN	TUOI	DIACHI	LUONG
1	Thanh	32	Haiphong	2000.00
2	Loan	25	Hanoi	1500.00
3	Nga	23	Hanam	2000.00
4	Manh	25	Hue	6500.00
5	Huy	27	Hatinh	8500.00
6	Cao	22	HCM	4500.00
7	Lam	24	Hanoi	10000.00

TT_ID	NGAY	NHANVIEN_ID	SOTIEN
102	2019-01-08 00:00:00	3	3000
100	2019-01-08 00:00:00	3	1500
101	2019-02-20 00:00:00	2	1560
103	2018-12-20 00:00:00	4	2060

➤ Join 2 bảng bằng LEFT JOIN

SELECT ID, TEN, SOTIEN, NGAY

FROM NHANVIEN

LEFT JOIN TIENTHUONG

ON NHANVIEN.ID =

TIENTHUONG.NHANVI

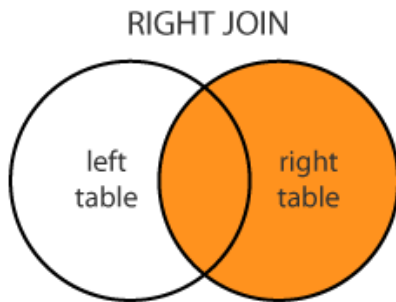
EN\_ID:

ID	TEN	SOTIEN	NGAY
1	Thanh	NULL	NULL
2	Loan	1560	2019-02-20 00:00:00
3	Nga	3000	2019-01-08 00:00:00
3	Nga	1500	2019-01-08 00:00:00
4	Manh	2060	2018-12-20 00:00:00
5	Huy	NULL	NULL
6	Cao	NULL	NULL
7	Lam	NULL	NULL

# RIGHT JOIN

## ❑ Mệnh đề RIGHT JOIN

- Mệnh đề **RIGHT JOIN** trong **SQL** là kiểu **JOIN** trả về tất cả các **bản ghi** từ **bảng** bên **PHẢI** (bảng 2) và các **bản ghi** phù hợp từ **bảng** bên **TRÁI** (bảng 1).
- Nếu mệnh đề **ON** không khớp với **bản ghi** nào trong **bảng** bên **trái** thì **RIGHT JOIN** sẽ vẫn trả về **một hàng** trong kết quả, nhưng giá trị là **NULL** trong mỗi **cột** từ **bảng** bên **trái**.



## ❖ Cú pháp:

```
SELECT cot1, cot2,... cotn  
FROM bang1  
RIGHT JOIN bang2  
ON bang1.cot_chung = bang2.cot_chung;
```

## ➤ Chú thích:

- ✓ **cot1, cot2,... cotn**: tên các cột cần hiển thị ở kết quả truy vấn. Các cot được ngăn cách với nhau bằng dấu phẩy (,)
- ✓ **bang1, bang2**: tên các bảng để lấy dữ liệu khi truy vấn.
- ✓ **cot\_chung**: thường là tên cột khóa ngoại tham chiếu từ bang1 đến cột định danh trong bang2 hoặc ngược lại.

# RIGHT JOIN

## ❑ Ví dụ

ID	TEN	TUOI	DIACHI	LUONG
1	Thanh	32	Haiphong	2000.00
2	Loan	25	Hanoi	1500.00
3	Nga	23	Hanam	2000.00
4	Manh	25	Hue	6500.00
5	Huy	27	Hatinh	8500.00
6	Cao	22	HCM	4500.00
7	Lam	24	Hanoi	10000.00

TT_ID	NGAY	NHANVIEN_ID	SOTIEN
102	2019-01-08 00:00:00	3	3000
100	2019-01-08 00:00:00	3	1500
101	2019-02-20 00:00:00	2	1560
103	2018-12-20 00:00:00	4	2060

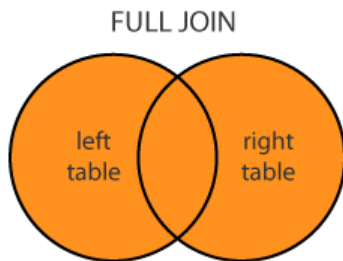
- JOIN hai bảng bằng RIGHT JOIN
- ```
SELECT ID, TEN, SOTIEN, NGAY
FROM NHANVIEN
RIGHT JOIN TIENTHUONG
ON NHANVIEN.ID =
TIENTHUONG.NHANVIEN_ID;
```
- ❖ Kết quả:

| ID | TEN  | SOTIEN | NGAY                |
|----|------|--------|---------------------|
| 3  | Nga  | 3000   | 2019-01-08 00:00:00 |
| 3  | Nga  | 1500   | 2019-01-08 00:00:00 |
| 2  | Loan | 1560   | 2019-02-20 00:00:00 |
| 4  | Manh | 2060   | 2018-12-20 00:00:00 |

# FULL JOIN

## ❑ Mệnh đề FULL JOIN

- Mệnh đề FULL JOIN trong SQL trả về tất cả bản ghi ở bảng trái và bảng phải kết hợp lại. Nói cách khác, mệnh đề này là kết hợp kết quả của cả hai loại LEFT và RIGHT JOIN.
- Bảng được kết hợp sẽ chứa tất cả bản ghi từ cả hai bảng và điền vào đó giá trị NULL cho các giá trị không khớp nhau.



## ❖ Cú pháp:

```
SELECT cot1, cot2,... cotn  
FROM bang1  
FULL JOIN bang2  
ON bang1.cot_chung = bang2.cot_chung;
```

## ➤ Chú thích:

- ✓ **cot1, cot2,... cotn**: tên các cột cần hiển thị ở kết quả truy vấn. Các cot được ngăn cách với nhau bằng dấu phẩy (,)
- ✓ **bang1, bang2**: tên các bảng để lấy dữ liệu khi truy vấn.
- ✓ **cot\_chung**: thường là tên cột khóa ngoại tham chiếu từ bảng đến cột định danh trong bảng hoặc ngược lại.



# FULL JOIN

## ❑ Ví dụ

| ID | TEN   | TUOI | DIACHI   | LUONG    |
|----|-------|------|----------|----------|
| 1  | Thanh | 32   | Haiphong | 2000.00  |
| 2  | Loan  | 25   | Hanoi    | 1500.00  |
| 3  | Nga   | 23   | Hanam    | 2000.00  |
| 4  | Manh  | 25   | Hue      | 6500.00  |
| 5  | Huy   | 27   | Hatinh   | 8500.00  |
| 6  | Cao   | 22   | HCM      | 4500.00  |
| 7  | Lam   | 24   | Hanoi    | 10000.00 |

| TT_ID | NGAY                | NHANVIEN_ID | SOTIEN |
|-------|---------------------|-------------|--------|
| 102   | 2019-01-08 00:00:00 | 3           | 3000   |
| 100   | 2019-01-08 00:00:00 | 3           | 1500   |
| 101   | 2019-02-20 00:00:00 | 2           | 1560   |
| 103   | 2018-12-20 00:00:00 | 4           | 2060   |

➤ JOIN hai bảng bằng FULL JOIN.

```
SELECT ID, TEN, SOTIEN, NGAY  
FROM NHANVIEN  
FULL JOIN TIENTHUONG  
ON NHANVIEN.ID =  
    TIENTHUONG.NHANVIEN
```

\_ID

| ID | TEN   | SOTIEN | NGAY                |
|----|-------|--------|---------------------|
| 1  | Thanh | NULL   | NULL                |
| 2  | Loan  | 1560   | 2019-02-20 00:00:00 |
| 3  | Nga   | 3000   | 2019-01-08 00:00:00 |
| 3  | Nga   | 1500   | 2019-01-08 00:00:00 |
| 4  | Manh  | 2060   | 2018-12-20 00:00:00 |
| 5  | Huy   | NULL   | NULL                |
| 6  | Cao   | NULL   | NULL                |
| 7  | Lam   | NULL   | NULL                |
| 3  | Nga   | 3000   | 2019-01-08 00:00:00 |
| 3  | Nga   | 1500   | 2019-01-08 00:00:00 |
| 2  | Loan  | 1560   | 2019-02-20 00:00:00 |
| 4  | Manh  | 2060   | 2018-12-20 00:00:00 |

## ❑ Truy vấn con (Subquery)

- Khi thực hiện một câu lệnh **SELECT** thì kết quả nó trả về một bảng tạm nên ta có thể viết câu truy vấn dạng **readonly** trên đó.
- Lệnh **Subquery** là cách viết một câu lệnh **SQL** mà trong đó có lồng thêm một hoặc nhiều câu truy vấn khác.
- Lệnh **Subquery** được nhúng trong mệnh đề **WHERE**.
- Thường được sử dụng trong bốn lệnh **SELECT**, **INSERT**, **UPDATE** hoặc **DELETE**.

## ❑ SUBQUERY với SELECT

### ❖ Cú pháp:

```
SELECT cot1, cot2,... cotn  
FROM tên_bảng  
WHERE tên_cột TOÁN_TỬ  
(SELECT cot1, cot2,... cotn  
FROM tên_bảng  
WHERE điều_kiện);
```

### ➤ Chú thích:

- ✓ **TOÁN\_TỬ**: Toán tử được sử dụng ở đây có thể là: =, <, >, >=, <=, IN, BETWEEN, v.v ...

# SUBQUERY SELECT

## ❑ Ví dụ

### ❖ Bảng CUSTOMER

| ID | NAME     | AGE | ADDRESS | SALARY   |
|----|----------|-----|---------|----------|
| 1  | Ha Anh   | 32  | Da Nang | 2000.00  |
| 2  | Van Ha   | 25  | Ha Noi  | 1500.00  |
| 3  | Vu Bang  | 23  | Vinh    | 2000.00  |
| 4  | Thu Minh | 25  | Ha Noi  | 6500.00  |
| 5  | Hai An   | 27  | Ha Noi  | 8500.00  |
| 6  | Hoang    | 22  | Ha Noi  | 4500.00  |
| 7  | Binh     | 24  | Ha Noi  | 10000.00 |

## ❑ Ví dụ

```
SELECT *  
FROM CUSTOMERS  
WHERE ID IN (SELECT ID  
FROM CUSTOMERS  
WHERE SALARY > 4500) ;
```

### ❖ Kết quả:

| ID | NAME     | AGE | ADDRESS | SALARY   |
|----|----------|-----|---------|----------|
| 4  | Thu Minh | 25  | Ha Noi  | 6500.00  |
| 5  | Hai An   | 27  | Ha Noi  | 8500.00  |
| 7  | Binh     | 24  | Ha Noi  | 10000.00 |

# SUBQUERY INSERT

## ❑ SUBQUERY với INSERT

### ❖ Cú pháp:

INSERT INTO tên\_bảng (cot1, cot2, ..., cotn)

SELECT cot1, cot2, ..., cotn

FROM tên\_bảng

WHERE tên\_cột TOÁN\_TỬ

( SELECT cot1, cot2, ..., cotn

FROM tên\_bảng

WHERE điều\_kiện );

### ➤ Chú thích:

- ✓ TOÁN\_TỬ: Toán tử được sử dụng ở đây có thể là: =, <, >, <=, >=, IN, BETWEEN, v.v ...

### ❖ Ví dụ:

- ✓ Sao chép toàn bộ thông tin từ bảng CUSTOMER vào bảng CUSTOMER\_COPY (có cấu trúc giống như bảng CUSTOMER);

INSERT INTO CUSTOMERS\_COPY

SELECT \* FROM CUSTOMERS

WHERE ID IN

( SELECT ID

FROM CUSTOMERS ) ;

# SUBQUERY UPDATE

## ❑ SUBQUERY với UPDATE

### ❖ Cú pháp:

UPDATE tên\_bảng

SET cot1 = giatri1, cot2 = giatri2, ..., cotn = giatriN

WHERE tên\_cột TOÁN\_TỬ

( SELECT cot1, cot2, ..., cotn

FROM tên\_bảng

WHERE điều\_kiện );

### ➤ Chú thích:

- ✓ **TOÁN\_TỬ**: Toán tử được sử dụng ở đây có thể là: =, <, >, >=, <=, IN, BETWEEN, v.v ...

### ❖ Ví dụ:

- ✓ Cập nhật lương tăng gấp 0.25 lần cho những khách hàng có tuổi lớn hơn 27.

UPDATE CUSTOMERS

SET SALARY = SALARY \* 0.25

WHERE AGE IN

( SELECT AGE

FROM CUSTOMERS

WHERE AGE >= 27 );

| ID | NAME     | AGE | ADDRESS | SALARY   |
|----|----------|-----|---------|----------|
| 1  | Ha Anh   | 32  | Da Nang | 125.00   |
| 2  | Van Ha   | 25  | Ha Noi  | 1500.00  |
| 3  | Vu Bang  | 23  | Vinh    | 2000.00  |
| 4  | Thu Minh | 25  | Ha Noi  | 6500.00  |
| 5  | Hai An   | 27  | Ha Noi  | 2125.00  |
| 6  | Hoang    | 22  | Ha Noi  | 4500.00  |
| 7  | Binh     | 24  | Ha Noi  | 10000.00 |

# SUBQUERY DELETE

## ❑ SUBQUERY với INSERT

### ❖ Cú pháp:

DELETE FROM tên\_bảng

WHERE tên\_cột TOÁN\_TỬ

( SELECT cot1, cot2, ..., cotn

FROM tên\_bảng

WHERE điều\_kiện );

### ➤ Chú thích:

- ✓ TOÁN\_TỬ: Toán tử được sử dụng ở đây có thể là: =, <, >, >=, <=, IN, BETWEEN, v.v ...

### ❖ Ví dụ:

- ✓ Xóa các bản ghi từ bảng CUSTOMER với những khách hàng có tuổi lớn hơn hoặc bằng 27.

DELETE FROM CUSTOMERS

WHERE AGE IN

( SELECT AGE

FROM CUSTOMERS

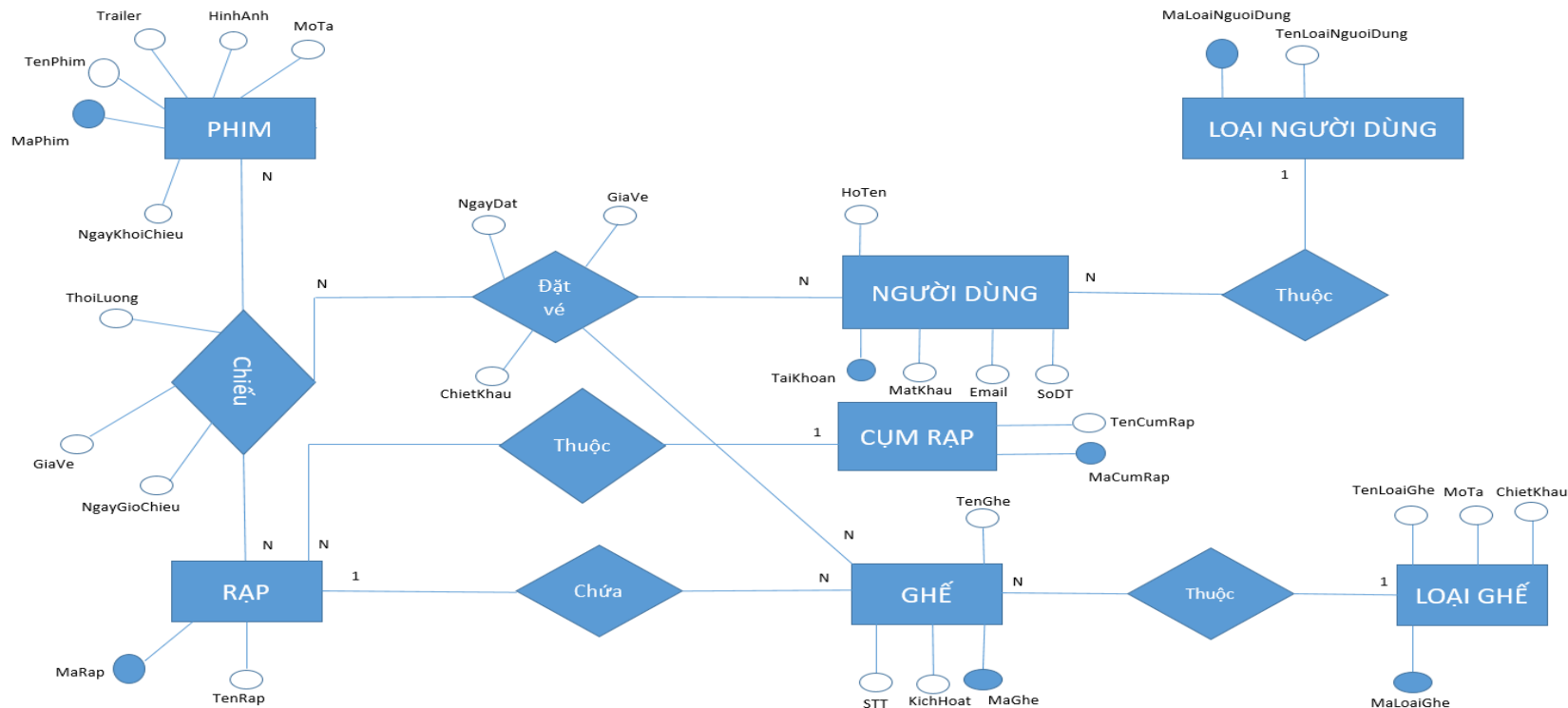
WHERE AGE >= 27 );

➤ K

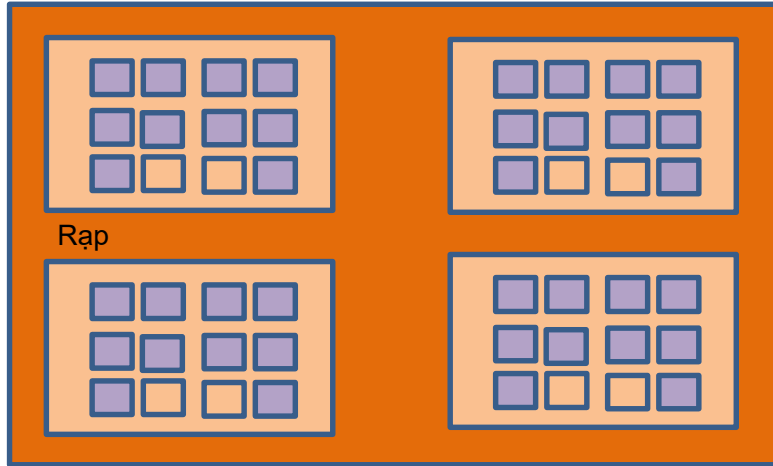
| ID | NAME     | AGE | ADDRESS | SALARY   |
|----|----------|-----|---------|----------|
| 2  | Van Ha   | 25  | Ha Noi  | 1500.00  |
| 3  | Vu Bang  | 23  | Vinh    | 2000.00  |
| 4  | Thu Minh | 25  | Ha Noi  | 6500.00  |
| 6  | Hoang    | 22  | Ha Noi  | 4500.00  |
| 7  | Binh     | 24  | Ha Noi  | 10000.00 |

# BÀI TẬP

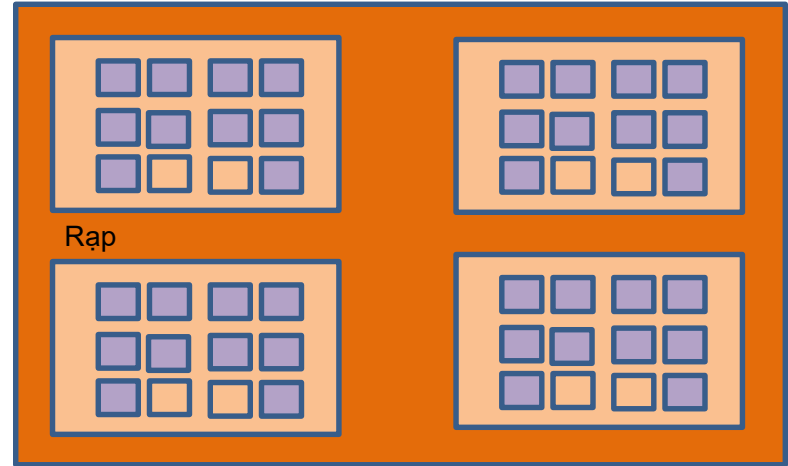
❑ Cho lược đồ như hình vẽ và cơ sở dữ liệu tương ứng



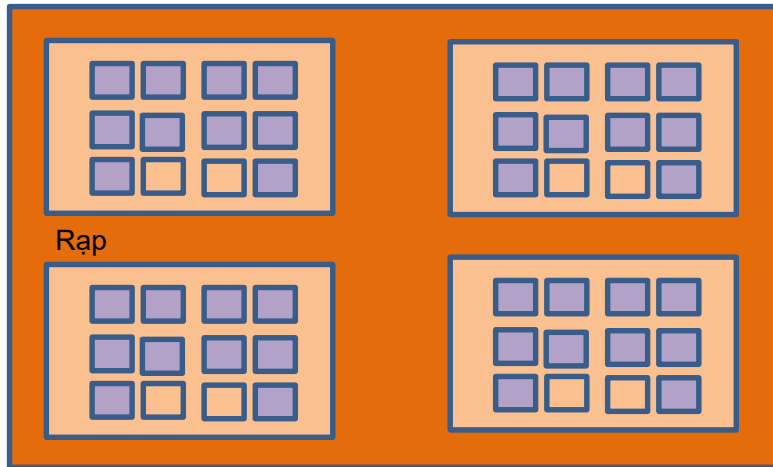
Cụm rập



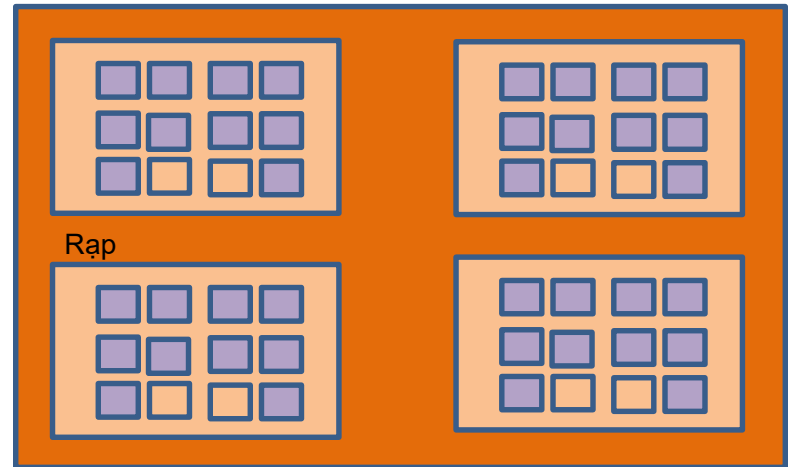
Cụm Rập



Cụm rập



Cụm rập





# Thực hiện truy vấn sau

1. Thực hiện câu truy vấn trả về các thông tin Rap(MaRap,TenRap, SoGhe,TenCumRap) có TenCumRap Mega GS
2. Thực hiện truy vấn trả về thông tin các Ghe(MaGhe, TenGhe, TenRap, STT, TenLoaiGhe, KichHoat) có TenLoaiGhe là Thường
3. Thực hiện truy vấn cho biết thông tin lịch chiếu của Phim (TenPhim, TenRap, NgayChieuGioChieu, GiaVe, ThoiLuong) của các phim có NgayChieuGioChieu = '2019-01-01 14:00:00'
4. Thực hiện truy vấn lấy thông tin người dùng (TaiKhoan,Email,SoDT) ở Nhóm 1
5. Thực hiện truy vấn trả về lịch chiếu của Phim có tên Phim chứa từ khóa Ted (TenPhim, TenRap, NgayChieuGioChieu, GiaVe, ThoiLuong)