



## TỔNG HỢP CODE - Tổng hợp các code dùng trong sql server trong môn quản trị cơ sở dữ liệu. KHÔNG

Quản trị Cơ sở dữ liệu (Trường Đại học Kinh tế - Tài chính Thành phố Hồ Chí Minh)

## TỔNG HỢP CODE

B1: Tạo database → B2: Tạo bảng → B3: Edit bảng/cột → B4: Tạo khóa 7749 cái → B5: thêm dữ liệu → B6: Edit dữ liệu → B7: Filter 1 bảng → B8: Filter nhiều bảng

Tạo database    **CREATE DATABASE <tên bảng>**

Xóa database    **DROP DATABASE <tên bảng>**

Chọn database **USE <tên bảng>**

Tạo bảng        **CREATE TABLE <tên bảng> (<các thuộc tính bên trong>)**

Xóa bảng        **DROP TABLE <tên bảng>**

Tạo column     **<tên thuộc tính> <kiểu dữ liệu> [<ràng buộc thuộc tính>]**

//Ràng buộc thuộc tính có thể là primary key, foreign key, constraint, not null

---

### Ràng buộc giá trị chọn – ADD CONSTRAINT

Cách 1: Trong lúc khai báo cột trong bảng:

**<tên cột> <kiểu dữ liệu>**

**CHECK (<tên cột> IN (<giá trị ràng buộc 1>, <giá trị ràng buộc 2>))**

Cách 2: Sau khi khai báo cột trong bảng – thêm điều kiện ràng buộc bên ngoài:

**ALTER TABLE <tên bảng> ADD CONSTRAINT <tên khóa điều kiện>**

**CHECK (<tên cột> IN (<giá trị ràng buộc 1>, <giá trị ràng buộc 2>))**

Hoặc

**ALTER TABLE <tên bảng> ADD CONSTRAINT <tên khóa điều kiện>**

**CHECK (<tên cột> = <giá trị ràng buộc 1> OR <tên cột> = <giá trị ràng buộc 2>))**

Ví dụ ràng buộc giá trị Nam Nữ cho giới tính:

Cách 1: **Phai char (3) CHECK (Phai IN ('Nam', 'Nu'))**

Cách 2: **ALTER TABLE NhanVien ADD CONSTRAINT CHECK (Phai IN ('Nam', 'Nu'))**

---

Thêm cột:        **ALTER TABLE <tên bảng> ADD <tên cột> <kiểu dữ liệu> [<RBTT>]**

Xóa cột:         **ALTER TABLE <tên bảng> DROP COLUMN <tên cột>**

Thay đổi kiểu dữ liệu / dung lượng bộ nhớ/ thêm ràng buộc null/not null

**ALTER TABLE <tên bảng> ALTER COLUMN <tên cột> <kiểu dữ liệu mới/dung lượng mới>**

Thêm ràng buộc thuộc tính – ràng buộc thuộc tính ở đây có thể hiểu là các khóa chính, khóa ngoại, các điều kiện dữ liệu dữ kiện tùy theo người dùng, nhưng cú pháp đều là đặt tên ràng buộc trước rồi mới cho biết loại ràng buộc là gì?

CHÚ Ý: set ràng buộc ở tại bảng thì không cần <tên RBTT> nhưng ở câu lệnh riêng thì phải có

**ALTER TABLE <tên bảng> ADD CONSTRAINT <Tên RBTT> <RBTT>**

Xóa ràng buộc thuộc tính:

**ALTER TABLE <tên bảng> DROP <Tên RBTT>**

Set khóa chính

```
ALTER TABLE <tên bảng> ADD CONSTRAINT <tên khóa> PRIMARY KEY (<tên thuộc tính>)
```

Set khóa ngoại:

```
ALTER TABLE <tên bảng> ADD CONSTRAINT <tên khóa> FOREIGN KEY <tên thuộc tính>  
REFERENCES <tên bảng> (<tên thuộc tính tham chiếu>)
```

Thêm ràng buộc

```
ALTER TABLE <tên bảng> ADD CONSTRAINT <tên khóa> CHECK (<các nội dung ràng buộc>)
```

VD:

```
//ràng buộc giá trị
```

```
ALTER TABLE nhanvien ADD CONSTRAINT gender CHECK (GT IN ('Nam', 'Nu'))
```

```
//ràng buộc giới hạn
```

```
ALTER TABLE nhanvien ADD CONSTRAINT age CHECKT (namsinh < 2000)
```

---

Thêm dữ liệu vào bảng: thêm theo từng cột thuộc tính hoặc thêm một loạt

Ngăn cách giữa các dòng giá trị record là dấu phẩy

```
INSERT INTO <tên bảng> (<thuộc tính 1>, thuộc tính 2>, ...)
```

```
VALUES      ('<Giá trị cho thuộc tính 1>', '<Giá trị cho thuộc tính 2>', ...),  
            ('<Giá trị cho thuộc tính 1>', '<Giá trị cho thuộc tính 2>', ...)
```

//Nhập hàng loạt theo thứ tự cột

```
INSERT INTO <tên bảng>
```

```
VALUES ('<Giá trị cho thuộc tính cột 1>', '<Giá trị cho thuộc tính 2>', ...)
```

Sửa dữ liệu trong bảng

```
UPDATE <tên bảng>
```

```
SET <tên cột 1> = <biểu thức hoặc giá trị sửa vào>, <tên cột 2> = <biểu  
thức hoặc giá trị sửa vào>, ...
```

//Dòng này để đổi dữ liệu trong cột

```
[WHERE] <Điều kiện 1> AND/OR <Điều kiện 2>
```

//Dòng này để giới hạn sửa có chọn lọc thay vì hàng loạt

VD: Đổi địa chỉ của tất cả nhân viên nữ thành HN và Phòng ban thành phòng số 3

```
UPDATE      NhanVien  
SET         DCHI = 'HN', PHG = 3  
WHERE       GT = N'Nữ'
```

Xóa một số hàng trong bảng: DELETE FROM <tên bảng> WHERE <điều kiện>

**LƯU Ý: THÊM TỪ CHÍNH SANG NGOẠI, XÓA TỪ NGOẠI SANG CHÍNH**

---

## TRUY VẤN DỮ LIỆU – FILTER

Cấu trúc chung: **SELECT** [**DISTINCT**] \* <tên cột> [<hàm>]

**FROM** <tên bảng>

[**WHERE** <điều kiện>]

[**GROUP BY** <tên cột>]

[**HAVING** <điều kiện>]

[**ORDER BY** <tên cột> **ASC/DESC**]

Nếu xuất toàn bộ các cột trong bảng thì viết: **SELETE \* FROM** <tên bảng>

Nếu chỉ xuất một số cột trong bảng thì ghi:

**SELETE** <thuộc tính 1>, <thuộc tính 2>, <thuộc tính 3> **FROM** <tên bảng>

Đổi tên hiển thị thuộc tính:

**SELECT** <tên thuộc tính> **as** [<Tên muốn hiển thị>] **FROM** <tên bảng>

Thể hiện nhiều giá trị của các cột thành 1 cột:

**SELECT** <tên thuộc tính cột nối 1> + 'space' + <tên thuộc tính cột nối thứ 2> + 'space' + <tên thuộc tính cột nối thứ 3> **as** 'Tên cột hiển thị của cột gộp'

**FROM** <tên bảng>

[**WHERE**] <Điều kiện>

Khử hàng trùng giá trị trong cột:

**SELECT DISTINCT** <tên thuộc tính muốn khử hàng trùng giá trị> **FROM** <tên bảng>

Các điều kiện lọc trong **WHERE**

- Toán tử so sánh: =, >, <, >=, <=, <>
- Toán tử logic: **AND**, **OR**, **NOT**
- Phép toán: +, -, \*, /
- **BETWEEN ... AND**: giới hạn vùng dữ liệu được dùng
  - o Vd: **LUONG > 2000 AND LUONG < 3000** tương đương với **LUONG BETWEEN 2000 AND 3000**
- **NOT BETWEEN**: giới hạn vùng dữ liệu cấm
- **IS (NOT) NULL**

VĐ: xuất nhân viên không có người quản lý

**SELECT** MaNV, TenNV **FROM** NHANVIEN

**WHERE** Ma\_NQL **IS NULL**

- **LIKE/ NOT LIKE**: So sánh chuỗi tương đối
  - o **Cú pháp**: %/\_ <chuỗi cần tìm> %/\_
    - % thay thế cho một chuỗi ký tự bất kỳ
    - \_ thay thế cho một ký tự bất kỳ
  - o VĐ: tìm hàng nào có chuỗi HCM: **LIKE ' %HCM'**
  - o VĐ: tìm nhân viên có mã có đuôi = 2: **LIKE '\_ \_ \_ 2'**

- (NOT) IN: WHERE <tên thuộc tính> IN (<giá trị 1>, <giá trị 2>, ...)  
  - o Tương đương với  
WHERE <tên thuộc tính> = <giá trị 1> OR <tên thuộc tính> = <giá trị 2>
- (NOT) EXISTS
- SOME, ALL
- Các hàm tính toán cơ bản:

#### □ Các hàm tính toán cơ bản

- **COUNT**: Đếm số bộ dữ liệu của thuộc tính
- **MIN**: Tính giá trị nhỏ nhất
- **MAX**: Tính giá trị lớn nhất
- **AVG**: Tính giá trị trung bình
- **SUM**: Tính tổng giá trị các bộ dữ liệu

□ Tính lương thấp nhất, cao nhất, trung bình và tổng lương của tất cả nhân viên

```
SELECT MIN(LUONG) AS [THẤP NHẤT],
       MAX(LUONG) AS [CAO NHẤT],
       AVG(LUONG) AS [TRUNG BÌNH],
       SUM(LUONG) AS [TỔNG LƯƠNG]
FROM NHANVIEN
```

	THẤP NHẤT	CAO NHẤT	TRUNG BÌNH	TỔNG LƯƠNG
1	1000000	5000000	2585714	18100000

Hiển thị theo hàm tính: SELECT <hàm tính> (<tên thuộc tính>) AS [<tên hiển thị>]

VD: Có tất cả bao nhiêu nhân viên: SELECT COUNT (\*) FROM NHANVIEN

VD: Bao nhiêu nhân viên có người quản lý:

```
SELECT COUNT (*) FROM NHANVIEN
WHERE Ma_NQL IS NOT NULL
```

Công thức:

```
SELECT COUNT/MIN/MAX/AVG/SUM <cột thuộc tính> * để lấy toàn bộ>
FROM <tên bảng> [WHERE <điều kiện>]
```

### TRUY VẤN DỮ LIỆU BẰNG PHÉP KẾT

Lấy mã và tên nhân viên thuộc phòng nghiên cứu:

```
SELECT n.MANV, n.TENNV // đặt lại tên cho chống trùng
FROM NHANVIEN n Join PHONGBAN p // Đặt lại tên bảng cho chống trùng
On n.PHG = p.MAPHONG // on trên cái gì là lấy so trên điều kiện đó
WHERE p.TENPHONG = N'NGHIÊN CỨU' // Lấy điều kiện lọc
```

Xuất ra họ tên nhân viên và tên phòng ban mà họ làm trưởng phòng

```
SELECT n.TENNV, p.TENPHONG
FROM NHANVIEN n Left Join PHONG BAN p On n.MANV = p.TRPHG
```

//Left / right join là để lấy dữ liệu dù có khớp không cũng đều hiển thị

Xuất họ tên NV và các đề án NV tham gia nếu có

```
SELECT d.TENDA, n.TENNV
FROM (DEAN d Join PHANCONG p On p.MADA = d.MADA)
Right Join NHANVIEN n On p.MA_NVNIEN = n.MANV
```

Với những đề án ở 'Hà Nội', cho biết mã đề án, mã phòng ban chủ trì đề án, họ tên trưởng phòng cùng với ngày sinh và địa chỉ của người ấy

```
SELECT      d.MADA, d.DDIEM_DA, d.PHONG, n.HOTEN, n.TENLOT, n.TENNV, n.NGSINH, n.DCHI
FROM        (DEAN d Join PHONGBAN p On d.PHONG = p.MAPHG)
            Join NHANVIEN n On n.MANV = p.TRPHG
WHERE       d.DDIEM_DA = 'Ha Noi'
```

Tìm họ tên của nhân viên phòng số 5 có tham gia vào đề án sản phẩm X với số giờ làm việc trên 10 giờ

```
SELECT      n.HONV, n.TENLOT, n.TENNV
FROM        (DEAN d Join PHANCONG p On d.MADA = p.MADA)
            Join NHANVIEN n On n.MANV = p.MA_NVIEN
WHERE       n.PHG = 5 And d.TENDA = 'San pham X' And p.THOIGIAN > 10
```

Tìm họ tên của từng nhân viên và người phụ trách trực tiếp nhân viên đó

```
SELECT      nv.TENNV As 'Tên NV', ql.TENNV As 'Tên QL'
FROM        NHANVIEN nv Join NHANVIEN ql On nv.MA_NQL = ql.MANV
```

---

TRUY VẤN GROUP BY – GROUP BY CÓ GÌ THÌ SELECT PHẢI CÓ CÁI ĐÓ!

Ví dụ: tính lương trung bình theo từng phòng ban

```
SELECT      PHG, AVG (LUONG) AS [LUONG TB]
FROM        NHANVIEN
GROUP BY    PHG                                     // Gộp theo thuộc tính phòng
```

VD 2: liệt kê tên đề án (TENDA) và tổng số giờ làm việc một tuần của tất cả nhân viên tham dự đề án đó

(Gộp theo tên đề án)

```
SELECT      d.TENDA, SUM (THOI GIAN)
FROM        DEAN d Join PHANCONG p On p.MADA = d.MADA
GROUP BY    d.TENDA                                //Đáng lý chỉ cần gom nhóm theo mã đề án nhưng mà khi hiển thị chỉ có tên
                                                    đề án mà không có mã đề án → hiển thị theo tên đề án
```

Cho biết họ và tên nhân viên và nhân viên đó có bao nhiêu người thân

```
SELECT      nv.HONV, nv.TENLOT, nv.TENNV, Count(TENTN)
FROM        NHANVIEN nv Join THANNHAN tn On nv.MANV = tn.MA_NVIEN
GROUP BY    nv.TENNV, nv.TENLOT, nv.HONV           //Group by sẽ theo thứ tự → cần trọng đặt thứ tự
```

TRUY VẤN HAVING – ĐIỀU KIỆN ĐÚNG SAU WHERE VÀ LỌC SAU KHI GROUP BY

VD: với các phòng ban có mức lương trung bình trên 30,000, liệt kê tên phòng ban và số lượng nhân viên của phòng ban đó → TB lương > 30,000 là điều kiện sau → Having

```
SELECT      p.TENPHG, Count(nv.MANV) as 'SLNV'
FROM        NHANVIEN nv Join PHONGBAN p On nv.PHG = p.MAPHG
GROUP BY    p.TENPHG
HAVING      AVG(nv.LUONG) > 30.000
```

## TRUY VẤN ORDER BY

VD: xuất ra 3 nhân viên có lương cao nhất

```
SELECT      TOP (<số lượng muốn xuất ra, ở đây là 3>) *
FROM        <tên bảng>
ORDER BY    <tên cột>
           //Mặc định là ASC (tăng dần), còn nếu ngược lại thì là DESC (giảm dần)
```

---

## TRUY VẤN LỒNG – TRUY VẤN CHA CON

Ví dụ: liệt kê ra các nhân viên không có thân nhân

**Cách 1: Dùng phép kết – Group by + Having**

```
SELECT      nv.MANV
FROM        NHANVIEN nv Left Join THANNHAN tn On nv.MANV = tn.MA_NVIEN
GROUP BY    MANV
HAVING      Count(MA_NVIEN) = 0
```

**Cách 2: Dùng truy vấn lồng**

```
SELECT      *
FROM        NHANVIEN
WHERE       MANV Not In ( SELECT      Distinct MA_NVIEN //Distinct để bỏ trùng
                        FROM        THANNHAN)
```

Cho biết các nhân viên có tham gia làm đề án “Thiết kế sản phẩm X”

```
SELECT      *
FROM        NHANVIEN n
WHERE       n.MANV = ( SELECT      p.MANVIEN
                        FROM        CONGVIEC c, PHANCONG p
                        WHERE       c.TEN_CONG_VIEC = N ‘Thiết kế sản phẩm X’
                        AND c.MADA = p.MADA
                        AND c.STT = p.STT)
```

Cho biết danh sách các đề án (MADA) có: nhân công với họ (HONV) là ‘Dinh’ hoặc có người trưởng phòng chủ trì đề án với họ (HONV) là ‘Dinh’

➔ Tách 2 vế:

- Truy vấn con
  - Bảng NHANVIEN làm điều kiện tham chiếu (WHERE): HONV = ‘Dinh’
  - Bảng PHONG BAN với bảng NHANVIEN bằng mã trưởng phòng với mã nhân viên
- Truy vấn cha – xuất ra các mã đề án yêu cầu

```

SELECT      Distinct p.MADA
FROM        PHANCONG p Right Join DEAN d On p.MADA = d.MADA
WHERE       p.MA_NVIEN In ( SELECT      nv.MANV
                              FROM        NHANVIEN nv
                              WHERE n.HONV = 'Dinh')
          Or d.PHONG In ( SELECT      p.MAPHG
                              FROM        PHONGBAN p Join NHANVIEN nv
                              On p.TRPHG = nv.MANV
                              WHERE nv.HONV = 'Dinh')

```

### Kiểu dữ liệu ký tự

Cú pháp kiểu dữ liệu	Kích thước tối đa	Giải thích
CHAR(kích_thuoc)	Tối đa 8000 kí tự.	<ul style="list-style-type: none"> <li>kích_thuoc là số kí tự lưu trữ.</li> <li>Độ dài cố định.</li> <li>Thêm dấu cách về bên phải để bù phần trống cho đủ số kí tự.</li> <li>Không chứa kí tự Unicode.</li> </ul>
VARCHAR(kích_thuoc) hoặc VARCHAR(toi_da)	Tối đa 8000 kí tự hoặc theo số tối đa.	<ul style="list-style-type: none"> <li>kích_thuoc là số kí tự lưu trữ.</li> <li>Độ dài tùy biến.</li> <li>Nếu chỉ định là toi_da thì tối đa là 2GB.</li> <li>Không chứa kí tự Unicode.</li> </ul>
TEXT	Tối đa 2GB.	<ul style="list-style-type: none"> <li>Độ dài tùy biến.</li> <li>Không chứa kí tự Unicode.</li> </ul>
NCHAR(kích_thuoc)	Tối đa 4000 kí tự.	<ul style="list-style-type: none"> <li>Độ dài cố định.</li> <li>Kí tự Unicode.</li> </ul>
NVARCHAR(kích_thuoc) hoặc NVARCHAR(toi_da)	Tối đa 4000 kí tự hoặc theo số tối đa.	<ul style="list-style-type: none"> <li>kích_thuoc là số kí tự lưu trữ.</li> <li>Độ dài tùy biến.</li> <li>Nếu số toi_da được chỉ định thì số kí tự tối đa là 2GB.</li> <li>Kí tự Unicode.</li> </ul>



NTEXT	Tối đa 1.073.741.823 byte.	<ul style="list-style-type: none"> <li>• Độ dài tùy biến.</li> <li>• Kí tự Unicode.</li> </ul>
BINARY(kích_thuoc)	Tối đa 8000 kí tự.	<ul style="list-style-type: none"> <li>• kích_thuoc là số kí tự lưu trữ.</li> <li>• Độ dài cố định.</li> <li>• Thêm dấu cách để bù phần trống cho đủ số kí tự.</li> <li>• Dữ liệu nhị phân.</li> </ul>
VARBINARY(kích_thuoc) hoặc VARBINARY(toi_da)	Tối đa 8000 kí tự hoặc theo số tối đa.	<ul style="list-style-type: none"> <li>• kích_thuoc là số kí tự lưu trữ.</li> <li>• Độ dài tùy biến.</li> <li>• Nếu chỉ định là toi_da thì tối đa là 2GB.</li> <li>• Dữ liệu nhị phân.</li> </ul>
IMAGE	kích thước tối đa là 2GB.	<ul style="list-style-type: none"> <li>• Độ dài tùy biến.</li> <li>• Dữ liệu nhị phân.</li> </ul>

### Kiểu dữ liệu số

Cú pháp kiểu dữ liệu	Kích thước tối đa	Giải thích
BIT	số nguyên 0, 1 hoặc NULL	
TINYINT	từ 0 đến 255	
SMALLINT	từ -32768 đến 32767	
INT	-2,147,483,648 đến 2,147,483,647	
BIGINT	từ -9,223,372,036,854,775,808 đến 9,223,372,036,854,775,807	
DECIMAL(m,d)	<ul style="list-style-type: none"> <li>• m mặc định là 18 nếu không được chỉ định cụ thể.</li> <li>• d mặc định là 0 nếu không được chỉ định cụ thể.</li> </ul>	m là tổng số lượng các số còn d là số lượng các số nằm sau dấu phẩy.
DEC(m,d)	<ul style="list-style-type: none"> <li>• m mặc định là 18 nếu không được chỉ định cụ thể.</li> <li>• d mặc định là 0 nếu không được chỉ định cụ thể.</li> </ul>	m là tổng số lượng các số còn d là số lượng các số nằm sau dấu phẩy. Đồng nghĩa với kiểu dữ liệu DECIMAL.

NUMERIC(m,d)	<ul style="list-style-type: none"> <li>m mặc định là 18 nếu không được chỉ định cụ thể.</li> <li>d mặc định là 0 nếu không được chỉ định cụ thể.</li> </ul>	<p>m là tổng số lượng các số còn d là số lượng các số nằm sau dấu phẩy.</p> <p>Đồng nghĩa với kiểu dữ liệu DECIMAL.</p>
FLOAT(n)	số dấu phẩy động n mặc định là 53 nếu không được chỉ định cụ thể.	n là số lượng của số bit lưu trữ trong một kí hiệu hóa học.
REAL	tương đương với FLOAT(24)	
SMALLMONEY	từ - 214,748.3648 đến 214,748.3647	
MONEY	từ -922,337,203,685,477.5808 đến 922,337,203,685,477.5807	

Kiểu dữ liệu ngày tháng năm

Cú pháp kiểu dữ liệu	Kích thước tối đa	Giải thích
DATE	giá trị từ '0001-01-01' đến '9999-12-31'.	hiển thị dưới dạng 'YYYY-MM-DD'
DATETIME	<ul style="list-style-type: none"> <li>Ngày lấy từ '1753-01-01 00:00:00' to '9999-12-31 23:59:59'.</li> <li>Giờ lấy từ '00:00:00' to '23:59:59:997'</li> </ul>	hiển thị dưới dạng 'YYYY-MM-DD hh:mm:ss[.mmm]
DATETIME2(chính xác tới số thập phân của giây)	<ul style="list-style-type: none"> <li>giá trị lấy từ '0001-01-01' đến '9999-12-31'.</li> <li>Thời gian lấy từ '00:00:00' đến '23:59:59:9999999'.</li> </ul>	hiển thị dưới dạng 'YYYY-MM-DD hh:mm:ss[.số giây thập phân]'
SMALLDATETIME	<ul style="list-style-type: none"> <li>giá trị lấy từ '1900-01-01' đến '2079-06-06'.</li> <li>Thời gian lấy từ '00:00:00' đến '23:59:59'.</li> </ul>	hiển thị dưới dạng 'YYYY-MM-DD hh:mm:ss'
TIME	<ul style="list-style-type: none"> <li>giá trị lấy từ '00:00:00.0000000' đến '23:59:59.9999999'.</li> <li>Ngày lấy từ '0001-01-01' đến '9999-12-31'.</li> </ul>	hiển thị dưới dạng 'YYYY-MM-DD hh:mm:ss[.nnnnnnnn]'
DATETIMEOFFSET(chính xác tới số thập phân của giây)	<ul style="list-style-type: none"> <li>giá trị thời gian lấy từ '00:00:00' đến '23:59:59:9999999'.</li> <li>Múi giờ lấy từ -14:00 đến +14:00.</li> </ul>	hiển thị dưới dạng 'YYYY-MM-DD hh:mm:ss[.nnnnnnnn] [{+ -}hh:mm]'

## FULL SET

```
//tạo database
CREATE DATABASE <tên database>
GO
USE <tên database>
GO

//tạo bảng
CREATE TABLE <tên bảng>
(
<thuộc tính 1>    <kiểu dữ liệu>    [<RBTV>],
<thuộc tính 2>    <kiểu dữ liệu>    [<RBTV>]
)
GO

//thêm cột
ALTER TABLE <tên bảng> ADD <tên cột> <kiểu dữ liệu> [<RBTV>]

//xóa cột
ALTER TABLE <tên bảng> DROP COLUMN <tên cột>

//thêm ràng buộc thuộc tính
ALTER TABLE <tên bảng> ADD CONSTRAINT <tên ràng buộc> <điều kiện ràng buộc>

//khóa chính
ALTER TABLE <tên bảng> ADD CONSTRAINT <tên khóa> PRIMARY KEY (<tên thuộc tính>)

//khóa ngoại
ALTER TABLE <tên bảng> ADD CONSTRAINT <tên khóa> FOREIGNER KEY (<tên thuộc tính>)
REFERENCES <tên bảng> (<tên thuộc tính tham chiếu>)
GO

//Truy vấn dữ liệu
SELECT      [DISTINCT] */<tên cột> [<hàm>]
FROM        <tên bảng> [JOIN <tên bảng nối> ON <điều kiện nối>]
[WHERE      <điều kiện>]
[GROUP BY  <tên cột>]          //cột nào có ở GROUP BY thì SELECT phải có!
[HAVING    <điều kiện>]       //loại điều kiện xuất hiện sau khi GROUP BY
[ORDER BY  <tên cột> ASC/DESC]
```