



# Quản Trị Dữ Liệu Với Microsoft SQL Server

## Chương: 7

### Tạo Bảng





# Mục tiêu

- Liệt kê các kiểu dữ liệu trong SQL Server 2012
- Mô tả các thủ tục tạo, sửa và xóa bảng trong csdl SQL Server
- Mô tả các thủ tục tạo, sửa và xóa cột trong của một bảng



# Giới thiệu

- Một trong các đối tượng csdl quan trọng nhất trong SQL Server 2012 là bảng.
- Bảng trong SQL Server 2012 chứa dữ liệu theo dạng dòng và cột.
- Mỗi cột có thể có chứa dữ liệu và kích thước xác định.

# Kiểu dữ liệu

Một kiểu dữ liệu là một thuộc tính xác định loại dữ liệu mà một đối tượng có thể lưu trữ, chẳng hạn như là dữ liệu số, dữ liệu kí tự, dữ liệu tiền tệ, vv..

Kiểu dữ liệu chỉ ra khả năng lưu trữ của một đối tượng.

Khi một cột đã được định nghĩa để lưu trữ dữ liệu thuộc về một kiểu dữ liệu cụ thể, thì các dữ liệu khác kiểu không thể lưu trữ trong nó được.

Như vậy, các kiểu dữ liệu cũng là một sự thực thi(enforce) toàn vẹn dữ liệu

Cho nên, nếu cố tình nhập dữ liệu kí tự vào một cột số nguyên, thì sẽ không thành công.

# Các loại kiểu dữ liệu khác nhau 1-6

- SQL Server 2012 hỗ trợ ba loại kiểu dữ liệu khác nhau:

## Kiểu dữ liệu hệ thống (System data types)

- Đây là các kiểu được cung cấp sẵn bởi SQL Server 2012.

## Kiểu dữ liệu bí danh (Alias data types)


- Đây là các kiểu dữ liệu dựa trên các kiểu dữ liệu hệ thống cung cấp.
- Một trong những ứng dụng điển hình của kiểu dữ liệu bí danh là khi có nhiều bảng, trong các bảng này đều có các cột có đặc điểm giống nhau như lưu trữ cùng một kiểu dữ liệu có chiều dài như nhau, cho phép null.
- Trong trường hợp này, có thể tạo ra một kiểu dữ liệu bí danh để có thể được dùng chung cho tất cả các bảng này.

## Kiểu dữ liệu do người dùng định nghĩa

- Những kiểu này được tạo bằng việc dùng các ngôn ngữ lập trình được hỗ trợ bởi .NET Framework, là một framework phần mềm được phát triển bởi Microsoft.

## Các loại kiểu dữ liệu khác nhau 2-6

- Bảng dưới đây trình bày các kiểu dữ liệu trong SQL Server 2012 cùng với nhóm và mô tả về chúng:

Category	Data Type	Description
Exact Numerics 	int	Represents a column that occupies 4 bytes of memory space. Is typically used to hold integer values.
	smallint	Represents a column that occupies 2 bytes of memory space. Can hold integer data from -32,768 to 32,767.
	tinyint	Represents a column that occupies 1 byte of memory space. Can hold integer data from 0 to 255.

# Các loại kiểu dữ liệu khác nhau 3-6

Category	Data Type	Description
Exact Numerics	bigint	Represents a column that occupies 8 bytes of memory space. Can hold data in the range $-2^{63}$ (-9,223,372,036,854,775,808) to $2^{63}-1$ (9,223,372,036,854,775,807)
	numeric	Represents a column of this type that fixes precision and scale.
	money	Represents a column that occupies 8 bytes of memory space. Represents monetary data values ranging from $(-2^{63}/10000)$ (-92,337,203,685,477.5808) through $2^{63}-1$ (922,337,203,685,477.5807).



# Các loại kiểu dữ liệu khác nhau 4-6

Category	Data Type	Description
Approximate Numerics	float	Represents a column that occupies 8 bytes of memory space. Represents floating point number ranging from $-1.79E+308$ through $1.79E+308$ .
	real	Represents a column that occupies 4 bytes of memory space. Represents floating precision number ranging from $-3.40E+38$ through $3.40E+38$ .
Date and Time	datetime	Represents date and time. Stored as two 4-byte integers.
	smalldatetime	Represents date and time.
Character String	char	Stores character data that is fixed-length and non-Unicode.
	varchar	Stores character data that is variable-length and non-Unicode.
	text	Stores character data that is variable-length and non-Unicode.
Unicode Types	nchar	Stores Unicode character data of fixed-length.
	nvarchar	Stores variable-length Unicode character data.



# Các loại kiểu dữ liệu khác nhau 5-6

Category	Data Type	Description
Other Data Types	Timestamp	Represents a column that occupies 8 bytes of memory space. Can hold automatically generated, unique binary numbers that are generated for a database.
	binary(n)	Stores fixed-length binary data with a maximum length of 8000 bytes.
Other Data Types	varbinary(n)	Stores variable-length binary data with a maximum length of 8000 bytes.
	image	Stores variable-length binary data with a maximum length of $2^{30}-1$ (1,073,741,823) bytes.
	uniqueidentifier	Represents a column that occupies 16 bytes of memory space. Also, stores a globally unique identifier (GUID).

# Các loại kiểu dữ liệu khác nhau 6-6

- Các kiểu dữ liệu bí danh được tạo bằng lệnh CREATE TYPE .
- Cú pháp lệnh CREATE TYPE như sau:

## Cú pháp:

```
CREATE TYPE[ schema_name.] type_name{FROM base_type[(  
precision[,scale])][NULL|NOT NULL]}[;]
```

Trong đó,

schema\_name: chỉ ra tên của lược đồ mà kiểu dữ liệu bí danh được tạo trong đó.

type\_name: chỉ ra tên kiểu dữ liệu bí danh được tạo.

base\_type: chỉ ra tên của kiểu dữ liệu hệ thống mà kiểu dữ liệu bí danh được tạo dựa trên nó.

precision và scale: chỉ ra độ chính xác và tỷ lệ(scale) cho dữ liệu kiểu số.

NULL|NOT NULL: chỉ ra kiểu dữ liệu có thể có giá trị null hay không có giá trị null.

- Đoạn code dưới đây trình bày cách tạo kiểu dữ liệu bí danh có tên **usertype** bằng lệnh:

```
CREATE TYPE usertype FROM varchar(20) NOT NULL
```

# Tạo bảng 1-2

- Sử dụng câu lệnh CREATE TABLE để tạo các bảng trong SQL Server 2012.
- Cú pháp của lệnh CREATE TABLE như sau:

## Cú pháp

```
CREATE TABLE [database_name. [schema_name]. |  
schema_name.] table_name  
([<column_name>] [data_type] Null/Not Null,)  
ON [filegroup | "default"]  
GO
```

Trong đó,

database\_name: là tên của csdl mà bảng sẽ được tạo trong nó.

table\_name: là tên của bảng được tạo mới. Tên bảng có thể chứa tối đa 128 kí tự.

column\_name: là tên của cột có trong bảng. Tên cột có thể có tối đa 128 kí tự. Với các cột được tạo có kiểu dữ liệu timestamp thì không chỉ ra tên cột. Tên mặc định của cột timestamp là timestamp.

data\_type: chỉ ra kiểu dữ liệu cho cột được tạo.



## Tạo bảng 2-2

- Đoạn mã dưới đây minh họa việc tạo bảng có tên **dbo.Customer\_1**:

```
CREATE TABLE [dbo].[Customer_1] (  
  [Customer_id number] [numeric](10, 0) NOT NULL,  
  [Customer_name] [varchar](50) NOT NULL)  
ON [PRIMARY]  
GO
```

# Chỉnh sửa bảng 1-2

- Câu lệnh ALTER TABLE được sử dụng để chỉnh sửa cấu trúc bảng như sửa, thêm, hoặc xóa các cột và các ràng buộc(constraints), phân lại phân vùng, hoặc vô hiệu(disabling) hay cho phép(enabling) các ràng buộc và trigger.
- Cú pháp lệnh ALTER TABLE như sau:

## Syntax:

```
ALTER TABLE [[database_name. [schema_name].| schema_name.]table_name  
ALTER COLUMN ([<column_name>] [data_type] Null/Not Null,);  
| ADD ([<column_name>] [data_type] Null/Not Null,);  
| DROP COLUMN ([<column_name>];
```

Trong đó,

ALTER COLUMN: chỉ ra cột cụ thể sẽ được sửa đổi hoặc chỉnh sửa.

ADD: chỉ ra một hoặc nhiều cột được bổ sung thêm vào bảng.

DROP COLUMN ([<column\_name>]: chỉ ra cột có tên column\_name sẽ bị xóa khỏi bảng.

## Chỉnh sửa bảng 2-2

- Đoạn code sau đây minh họa sửa cột **Customer\_id**:

```
USE [CUST_DB]
ALTER TABLE [dbo].[Customer_1]
ALTER Column [Customer_id number] [numeric](12, 0) NOT NULL;
```

- Đoạn code sau đây minh họa việc bổ sung thêm cột **Contact\_number**:

```
USE [CUST_DB]
ALTER TABLE [dbo].[Table_1]
ADD [Contact_number] [numeric](12, 0) NOT NULL;
```

- Đoạn code sau đây minh họa xóa cột **Contact\_number**:

```
USE [CUST_DB]
ALTER TABLE [dbo].[Table_1]
DROP COLUMN [Contact_name];
```

- Có một số tình huống các cột không thể xóa như: nếu chúng có sử dụng ràng buộc CHECK, FOREIGN KEY, UNIQUE, hay PRIMARY KEY, gắn với định nghĩa DEFAULT, vv...

# Xóa bảng

- Câu lệnh `DROP TABLE` được dùng để xóa một bảng, dữ liệu của nó, và các đối tượng gắn với bảng đó như indexes, triggers, constraints, và permission.
- Cú pháp lệnh `DROP TABLE` statement như sau:

## Cú pháp

```
DROP TABLE <Table_Name>
```

Trong đó,

<Table\_Name>: Tên của bảng cần xóa.

- Đoạn code dưới đây minh họa cách xóa một bảng:

```
USE [CUST_DB]  
DROP TABLE [dbo].[Table_1]
```



# Các câu lệnh sửa đổi dữ liệu 1-4

- Các câu lệnh được sử dụng để sửa đổi dữ liệu là INSERT, UPDATE, và DELETE.

## Câu lệnh INSERT

- Câu lệnh INSERT thêm một dòng mới vào bảng
- Cú pháp câu lệnh như sau :

## Cú pháp

```
INSERT [INTO] <Table_Name> VALUES <values>
```

Trong đó,

<Table\_Name>: là tên của bảng mà các dòng sẽ được chèn vào.

[INTO] : là từ khóa tùy chọn được đặt ở giữa INSERT và bảng đích.

<Values>: chỉ ra các giá trị cho cột của bảng.

# Các câu lệnh sửa đổi dữ liệu 2-4

- Đoạn code dưới đây minh họa thêm các dòng vào bảng **Table\_2**:

```
USE [CUST_DB]
INSERT INTO [dbo].[Table_2] VALUES (101, 'Richard Parker', 'Richy')
GO
```

- Kết quả của lệnh trên là một dòng sẽ được chèn vào bảng.

## Câu lệnh UPDATE

- Câu lệnh UPDATE dùng để sửa dữ liệu trong bảng.
- Cú pháp lệnh UPDATE như sau:

## Cú pháp

```
UPDATE <Table_Name>
SET <Column_Name = Value>
[WHERE <Search condition>]
```

Trong đó,

<Table\_Name>: là tên của bảng mà các bản ghi sẽ được cập nhật.

<Column\_Name>: là tên của cột trong bảng mà các bản ghi được cập nhật tại đó.

# Các câu lệnh sửa đổi dữ liệu 3-4

<Value>: chỉ ra các giá trị mới cho các cột để chỉnh sửa.

<Search condition>: chỉ ra điều kiện các dòng được chỉnh sửa.

- Đoạn code dưới đây minh họa sử dụng câu lệnh UPDATE để chỉnh sửa giá trị trong cột **Contact\_number**:

```
USE [CUST_DB]
UPDATE [dbo].[Table_2] SET Contact_number = 5432679 WHERE Contact_name
LIKE 'Richy'
GO
```

- Hình sau cho thấy kết quả của câu lệnh UPDATE:

Results		Messages		
	Customer_id number	Customer_name	Contact_name	Contact_number
1	101	Richard Parker	Richy	5432679

# Các câu lệnh sửa đổi dữ liệu 4-4

## Câu lệnh DELETE

- Câu lệnh DELETE xóa bỏ các dòng khỏi bảng.
- Cú pháp câu lệnh DELETE như sau:

### Cú pháp

```
DELETE FROM <Table_Name>  
[WHERE <Search condition>]
```

Trong đó,

<Table\_Name>: là tên của bảng mà các dòng sẽ bị xóa khỏi nó.

Mệnh đề WHERE được sử dụng để chỉ ra điều kiện xóa. Nếu mệnh đề WHERE không được sử dụng, tất cả các dòng trong bảng sẽ bị xóa.

- Đoạn mã sau đây minh họa cách xóa các dòng có **Contact\_number** là **5432679** khỏi bảng **Customer\_2**:

```
USE [CUST_DB]  
DELETE FROM [dbo].[Customer_2] WHERE Contact_number = 5432679  
GO
```

# Cột cho phép (Column Nullability) 1-2

Đặc tính cho phép null của một cột là để xác định các dòng trong bảng có chứa giá trị null tại cột đó hay không.

Trong SQL Server, một giá trị null khác với 0(zero), hoặc để trống, hoặc độ dài chuỗi bằng 0 (như ' '). Ví dụ: giá trị null trong cột **Color** của bảng **Production**.

Bảng **Product** của csdl **AdventureWorks2012** không có nghĩa là sản phẩm không có màu; nó có nghĩa là không biết màu sản phẩm hoặc chưa xác định màu.

Cột cho phép null có thể được định nghĩa ngay lúc đang tạo bảng hoặc lúc chỉnh sửa bảng.

Từ khóa **NULL** được sử dụng để chỉ ra rằng các giá trị null được cho phép chứa trong cột và từ khóa **NOT NULL** để chỉ ra cột không cho phép chứa giá trị null.

## Cột cho phép (Column Nullability) 2-2

Khi chèn một dòng vào bảng mà không chỉ ra giá trị cho cột cho phép null thì SQL Server sẽ tự động đặt giá trị null vào cột đó, nếu như cột này không sử dụng DEFAULT để định nghĩa giá trị mặc định.

Cũng có thể nhập tường minh một giá trị null vào cột, bất kể cột đó có kiểu dữ liệu là gì hay có dùng giá trị mặc định đi chăng nữa.

- Trong đoạn mã dưới đây, lệnh `CREATE TABLE` sử dụng từ khóa `NULL` và `NOT NULL` cùng với định nghĩa cột:

```
USE [CUST_DB]
CREATE TABLE StoreDetails
(
    StoreID int NOT NULL,
    Name varchar(40) NULL
)
GO
```

- Kết quả của đoạn mã là bảng **StoreDetails** được tạo với các cột **StoreID** và **Name** được thêm vào bảng.

## Định nghĩa DEFAULT 1-3

Một định nghĩa DEFAULT được sử dụng để gán một giá trị mặc định nếu như không có giá trị nào được chỉ ra cho cột.

Ví dụ, thường dùng giá trị 0 làm giá trị mặc định cho các cột kiểu số, hoặc 'N/A' hay 'Unknown' làm mặc định cho các cột kiểu chuỗi (string) khi không có giá trị nào được chỉ ra.

Một định nghĩa DEFAULT có thể được tạo một cột tại thời điểm tạo bảng hoặc được thêm vào sau khi bảng đã tồn tại.

Nếu có một DEFAULT được thêm vào cột hiện có của bảng, SQL Server chỉ áp dụng giá trị mặc định mới cho các dòng được thêm mới.



## Định nghĩa DEFAULT 2-3

- Đoạn code dưới đây, câu lệnh CREATE TABLE sử dụng từ khóa DEFAULT để định nghĩa giá trị mặc định cho cột Price:

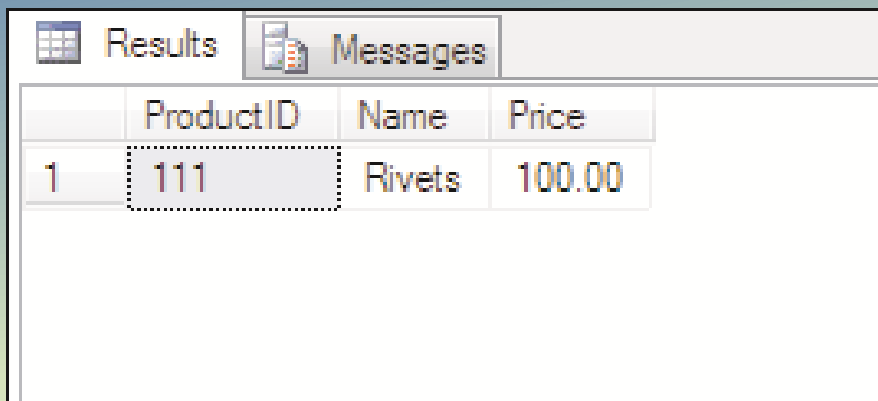
```
USE [CUST_DB]
CREATE TABLE StoreProduct
(
    ProductID int NOT NULL,
    Name varchar(40) NOT NULL,
    Price money NOT NULL DEFAULT (100)
)
GO
```

- Khi một dòng được thêm vào bảng bằng câu lệnh trong đoạn code dưới đây, giá trị của cột **Price** sẽ không bị để trống; nó sẽ có giá trị là **100.00** mặc dù người dùng không nhập bất kỳ giá trị nào cho cột đó.

```
USE [CUST_DB]
INSERT INTO dbo.StoreProduct (ProductID, Name) VALUES (111, 'Rivets')
GO
```

## Định nghĩa DEFAULT 3-3

- Hình dưới đây trình bày kết quả, trong đó các giá trị chỉ được thêm cho các cột **ProductID** và **Name**, nhưng cột **Price** vẫn có giá trị **100.00**.
- Là có một DEFAULT được định nghĩa trên cột.



	ProductID	Name	Price
1	111	Rivets	100.00

- Không thể tạo DEFAULT cho các cột sau đây:

Có kiểu dữ liệu `timestamp`

Có thuộc tính `IDENTITY` hoặc `ROWGUIDCOL`

Hiện đang tồn tại một định nghĩa `default` hoặc một đối tượng `default` rồi

# Thuộc tính IDENTITY 1-4

- Thuộc tính IDENTITY của SQL Server được sử dụng để tạo ra các cột định danh, chúng chứa các giá trị tự động phát sinh tuần tự để nhận dạng duy nhất mỗi hàng trong một bảng.
- Một cột định danh thường được sử dụng là khóa chính (primary key). Các đặc điểm của thuộc tính IDENTITY như sau:

Cột có thuộc tính IDENTITY phải là các cột có kiểu dữ liệu : decimal, int, numeric, smallint, bigint, hoặc tinyint.

Cột có thuộc tính IDENTITY không cần phải chỉ ra giá trị ban đầu(seed) và giá trị tăng(increment). Nếu không chỉ ra, giá trị 1 được sử dụng cho cả hai.

Mỗi bảng chỉ có duy nhất có một sử dụng thuộc tính IDENTITY

Cột định danh trong bảng phải là cột không cho phép null và không có chứa định nghĩa hoặc đối tượng DEFAULT.

Các cột được định nghĩa với thuộc tính IDENTITY thì giá trị của chúng không thể cập nhật được.

Các giá trị có thể được chèn tường minh cho cột identity nếu tùy chọn IDENTITY\_INSERT được thiết lập là ON.

Khi tùy chọn IDENTITY\_INSERT là ON, câu lệnh INSERT phải cung cấp giá trị cho cột identity.

## Thuộc tính IDENTITY 2-4

- Khi thuộc tính IDENTITY được thiết lập, việc lấy các giá trị của cột định danh có thể thực hiện bằng việc sử dụng từ khóa IDENTITYCOL với tên của bảng trong câu lệnh SELECT.
- Sử dụng hàm OBJECTPROPERTY ( ) để biết xem bảng có cột IDENTITY không.
- Sử dụng hàm COLUMNPROPERTY để lấy tên của cột IDENTITY có trong bảng.
- Cú pháp thêm thuộc tính IDENTITY trong lúc tạo bảng như sau:

### Cú pháp

```
CREATE TABLE <table_name> (  
column name data_type [ IDENTITY [(seed_value, increment_value)]  
NOT NULL  
)
```

Trong đó,

seed\_value: là giá trị ban đầu để bắt đầu phát sinh các giá trị định danh.

increment\_value: là giá trị tăng để tăng cho mỗi dòng mới được thêm vào bảng.

## Thuộc tính IDENTITY 3-4

- Đoạn code dưới đây minh họa sử dụng thuộc tính IDENTITY:

```
USE [CUST_DB]
GO
CREATE TABLE HRContactPhone (
    Person_ID int IDENTITY(500,1) NOT NULL,
    MobileNumber bigint NOT NULL
)
GO
```

- Tạo bảng **HRContactPhone** trong csdl **CUST\_DB** với hai cột.
- Cột **Person\_ID** là cột định danh.
- Giá trị ban đầu là **500**, và giá trị tăng là **1**.
- Khi chèn một dòng vào bảng, nếu tùy chọn **IDENTITY\_INSERT** không được bật là on thì phải chỉ ra giá trị tường minh cho cột IDENTITY.

## Thuộc tính IDENTITY 4-4

- Dưới đây là các lệnh minh họa chèn thêm một dòng vào bảng có cột Identity:

```
USE [CUST_DB]
INSERT INTO HRContactPhone (MobileNumber) VALUES (983452201)
INSERT INTO HRContactPhone (MobileNumber) VALUES (993026654)
GO
```

- Hình dưới đây trình bày kết quả, trong đó thuộc tính IDENTITY làm tăng các giá trị trong cột **Person\_ID**:

Results			Messages		
	Person_ID	MobileNumber			
1	500	983452201			
2	501	993026654			

# Cột định danh duy nhất toàn cầu 1-3

Ngoài thuộc tính `IDENTITY`, SQL Server còn hỗ trợ các định danh duy nhất tổng thể (globally unique identifiers).

Chỉ có thể tạo một cột định danh và một cột định danh duy nhất tổng thể cho mỗi bảng.

Để tạo và làm việc với cột nhận dạng duy nhất tổng thể, chúng ta phải sử dụng kết hợp từ khóa `ROWGUIDCOL`, kiểu dữ liệu `uniqueidentifier` và hàm `NEWID`

Các giá trị cho cột duy nhất tổng thể không được phát sinh tự động.

Phải tạo một định nghĩa `DEFAULT` với giá trị mặc định là hàm `NEWID()` cho cột `uniqueidentifier` để phát sinh giá trị duy nhất tổng thể (global unique).



# Cột định danh duy nhất toàn cầu 2-3

**Hàm** NEWID ( ) tạo một số định danh duy nhất là một chuỗi nhị phân 16 byte

**Cột có thể được tham chiếu đến trong phần danh sách cột của câu lệnh SELECT bằng từ khóa ROWGUIDCOL.**

**Sử dụng hàm OBJECTPROPERTY để biết bảng nào có cột ROWGUIDCOL.**

**Hàm COLUMNPROPERTY được dùng để lấy tên của cột ROWGUIDCOL.**

## Cột định danh duy nhất toàn cầu 3-3

- Đoạn code dưới đây minh họa dùng câu lệnh CREATE TABLE tạo bảng **EMPCellularPhone**.
- Cột **Person\_ID** phát sinh GUID tự động cho mỗi dòng mới được thêm vào bảng.

```
USE [CUST_DB]
CREATE TABLE EMP_CellularPhone
( Person_ID uniqueidentifier DEFAULT NEWID() NOT NULL,
  PersonName varchar(60) NOT NULL
)
GO
```

- Đoạn code dưới đây thêm giá trị cho cột **PersonName** column:

```
USE [CUST_DB]
INSERT INTO EMP_CellularPhone(PersonName) VALUES ('William Smith')
SELECT * FROM EMP_CellularPhone
GO
```

- Hình dưới đây hiển thị kết quả, trong đó một bộ định danh duy nhất được hiển thị ở cột **PersonName**:

Results		Messages	
Person_ID		PersonName	
1	362C4377-D194-4607-A466-7FF02064EAFD	William Smith	

# Các ràng buộc (constraint)

- Một ràng buộc là một thuộc tính được gắn tới một cột hoặc một tập các cột trong bảng để ngăn chặn các kiểu giá trị dữ liệu nào đó không nhất quán được nhập vào bảng.

Một ràng buộc(constraint) là một thuộc tính được gắn tới cột để áp dụng các nguyên tắc xử lý logic (business logic rule) và đảm bảo tính toàn vẹn.

Các ràng buộc có thể tạo ra ngay khi sử dụng câu lệnh CREATE TABLE để tạo bảng, hoặc cũng có thể được thêm vào lúc chỉnh sửa bảng bằng câu lệnh ALTER TABLE

Một ràng buộc mức cột có thể được chỉ ra như là một phần của định nghĩa cột và chỉ áp dụng tới cột đó.

Một ràng buộc mức bảng có thể sử dụng khi muốn áp dụng cho nhiều hơn một bảng và khai báo độc lập khỏi phần định nghĩa cột.

Hãy sử dụng các ràng buộc mức bảng khi có một hay nhiều cột sử dụng cùng một ràng buộc.

- SQL Server hỗ trợ các loại ràng buộc sau:
  - PRIMARY KEY
  - UNIQUE
  - FOREIGN KEY
  - CHECK
  - NOT NULL

# Khoá chính - PRIMARY KEY 1-3

Mỗi bảng thường có một khóa chính gồm một cột hay kết hợp nhiều cột để xác định duy nhất mỗi hàng bên trong bảng.

Ràng buộc PRIMARY KEY được sử dụng để tạo một khóa chính và đảm bảo toàn vẹn thực thể của bảng.

Mỗi bảng chỉ được tạo duy nhất một ràng buộc khóa chính.

Hai dòng trong cùng bảng không thể có cùng giá trị khóa chính và cột khóa chính không nhận các giá trị NULL.

➤ Cú pháp để thêm một khóa chính trong khi tạo một bảng:

```
CREATE TABLE <table_name>
(
    Column_name datatype PRIMARY KEY [column_list]
)
```

## Khoá chính - PRIMARY KEY 2-3

- Đoạn code dưới đây minh họa cách tạo bảng **EMPContactPhone** để lưu trữ thông tin điện thoại liên lạc của một người.
- Do cột **EMP\_ID** phải là khóa chính để xác định mỗi dòng là duy nhất, nên nó được tạo với một ràng buộc primary key.

```
USE [CUST_DB]
CREATE TABLE EMPContactPhone
(   EMP_ID int PRIMARY KEY,
    MobileNumber bigint,
    ServiceProvider varchar(30),
    LandlineNumber bigint
) GO
```

- Một phương pháp sử dụng khác từ khoá CONSTRAINT. Cú pháp như sau:

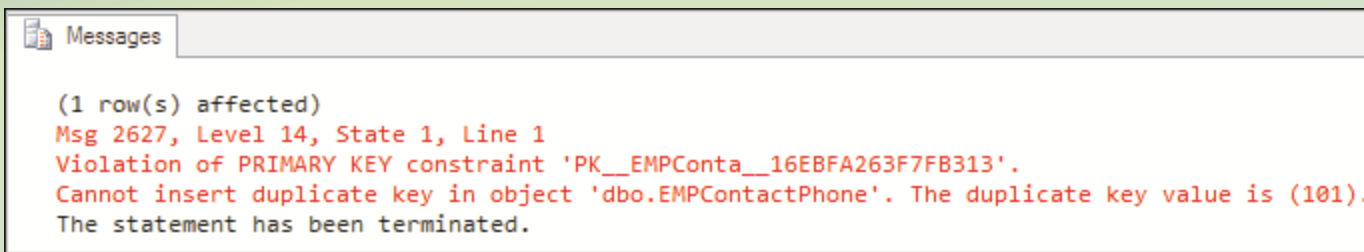
```
CREATE TABLE <table_name>
( <column name> <datatype> [, column_list] CONSTRAINT
  constraint_name PRIMARY KEY
)
```

# Khoá chính - PRIMARY KEY 3-3

- Đoạn code dưới đây là các câu lệnh truy vấn chèn các dòng dữ liệu cho bảng

```
USE [CUST_DB]
INSERT INTO dbo.EMPContactPhone values (101, 983345674, 'Hutch', NULL)
INSERT INTO dbo.EMPContactPhone values (102, 989010002, 'Airtel', NULL)
GO
```

- Câu lệnh đầu tiên trong đoạn code được thực thi thành công nhưng câu lệnh INSERT tiếp theo sẽ nhận được thông báo lỗi như hình dưới đây do giá trị cho cột EMP\_ID bị trùng :



- Hình dưới đây cho thấy kết quả sau khi thực hiện:

The screenshot shows the 'Results' window in SQL Server. It displays a table with the following data:

	EMP_ID	MobileNumber	ServiceProvider	LandlineNumber
1	101	983345674	Hutch	NULL

# UNIQUE 1-2

- Một ràng buộc UNIQUE được sử dụng để đảm bảo các giá trị được nhập vào một cột hoặc nhóm cột phải duy nhất (không trùng lặp).
- Ràng buộc duy nhất cho phép giá trị null.
- Một bảng có thể có nhiều hơn một ràng buộc UNIQUE.
- Cú pháp tạo ràng buộc UNIQUE như sau:

```
CREATE TABLE <table_name>
(
  [column_list ] <column_name> <data_type> UNIQUE [
  column_list]
)
```

- Đoạn code dưới đây minh họa cách tạo các cột UNIQUE là **MobileNumber** và **LandlineNumber**:

```
USE [CUST_DB]
GO
CREATE TABLE EMP_ContactPhone (
    Person_ID int PRIMARY KEY,
    MobileNumber bigint UNIQUE,
    ServiceProvider varchar(30),
    LandlineNumber bigint UNIQUE
)
```

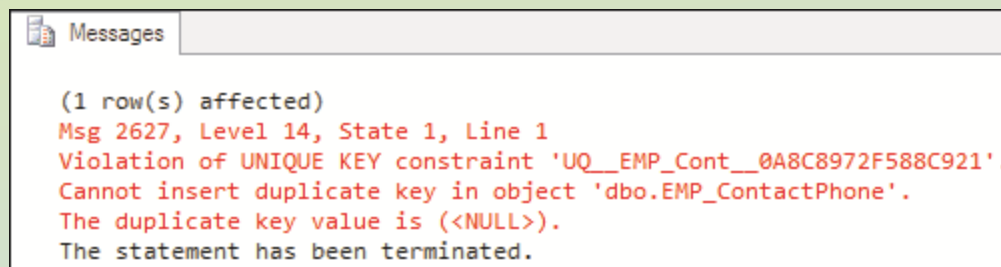


## UNIQUE 2-2

- Đoạn code dưới đây minh họa chèn các dòng dữ liệu vào bảng:

```
USE [CUST_DB]
INSERT INTO EMP_ContactPhone values (111, 983345674, 'Hutch', NULL)
INSERT INTO EMP_ContactPhone values (112, 983345674, 'Airtel', NULL)
GO
```

- Các ràng buộc UNIQUE chỉ kiểm tra tính duy nhất của các giá trị, chứ không ngăn cản việc nhập vào giá trị null.
- Lệnh INSERT thứ hai sẽ nhận được thông báo lỗi như hình dưới đây do giá trị cho cột **MobileNumber** bị trùng nhau:



- Nhận được lỗi do cột **MobileNumber** được định nghĩa là duy nhất và không cho phép trùng giá trị. Xem kết quả sau khi thực hiện trong hình dưới đây:

	Person_ID	MobileNumber	ServiceProvider	LandlineNumber
1	111	983345674	Hutch	NULL

# Khóa ngoại - FOREIGN KEY 1-2

- Khóa ngoại trong một bảng là một cột mà chỉ đến một khóa chính trong một bảng khác.
- Ràng buộc khóa ngoại (Foreign key) được sử dụng để đảm bảo toàn vẹn tham chiếu.
- Cú pháp tạo khóa ngoại:

```
CREATE TABLE <table_name>(  
[ column_list,] <column_name> <datatype>  
FOREIGN KEY REFERENCES <table_name> (pk_column_name> [, column_list]  
)
```

Trong đó,

table\_name: là tên của bảng mà tham chiếu khóa chính từ đó.

<pk\_column\_name>: là tên của cột khóa chính.

- Đoạn code dưới đây minh họa cách tạo ràng buộc khóa ngoại :

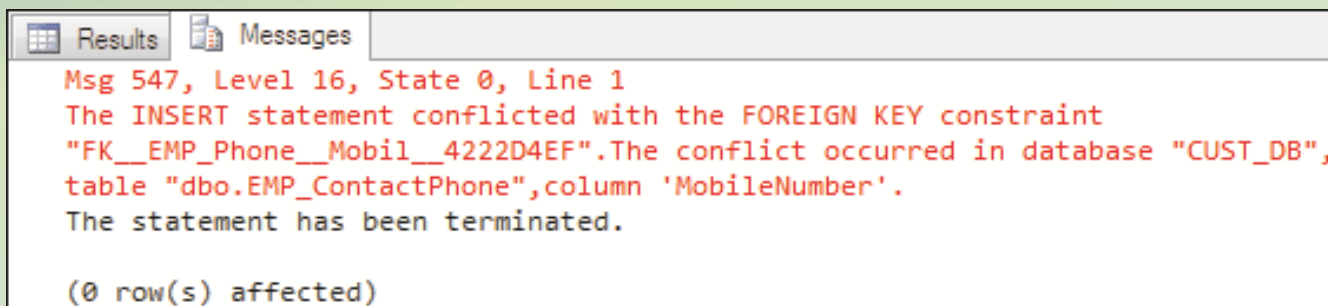
```
USE [CUST_DB]  
GO  
CREATE TABLE EMP_PhoneExpenses (  
Expense_ID int PRIMARY KEY,  
MobileNumber bigint FOREIGN KEY REFERENCES EMP_ContactPhone  
(MobileNumber), Amount bigint)
```

## Khóa ngoại - FOREIGN KEY 2-2

- Khi chèn một dòng vào bảng **EMP\_PhoneExpenses**, thì số di động(mobile number) phải có trong cột MobileNumber của bảng **EMP\_ContactPhone**.
- Đoạn code dưới đây minh họa chèn một dòng dữ liệu vào bảng **EMP\_PhoneExpenses** :

```
INSERT INTO dbo.EMP_PhoneExpenses values(101, 993026654, 500)
SELECT * FROM dbo.EMP_PhoneExpenses
```

- Đoạn mã trên nhận được thông báo lỗi như hình dưới đây:



- Nếu chèn một giá trị vào cột khóa ngoại mà giá trị này không có trong cột được tham chiếu đến, việc chèn sẽ bị lỗi như trong hình.
- Tuy nhiên là có thể thêm giá trị NULL vào cột khóa ngoại.

# CHECK 1-2

- Ràng buộc CHECK được sử dụng để giới hạn các giá trị có thể được nhập vào một cột.
- Ràng buộc CHECK đảm tính toàn vẹn của dữ liệu.
- Ràng buộc CHECK hoạt động bằng cách chỉ ra điều kiện tìm, là biểu thức có thể định giá(evaluate) TRUE, FALSE, hoặc không xác định (unknown).
- Các giá trị mà định giá(evaluate) là FALSE sẽ bị từ chối (rejected).
- Có thể chỉ ra nhiều ràng buộc CHECK cho một cột.
- Một ràng buộc CHECK cũng có thể áp dụng cho nhiều cột bằng cách tạo nó ở mức bảng.
- Đoạn code dưới đây minh họa tạo ràng buộc CHECK để đảm bảo giá trị cho **Amount** luôn phải lớn hơn 10:

```
USE [CUST_DB]
CREATE TABLE EMP_PhoneExpenses (
Expense_ID int PRIMARY KEY,
  MobileNumber bigint FOREIGN KEY REFERENCES EMP_ContactPhone
(MobileNumber),
Amount bigint CHECK (Amount >10))
GO
```

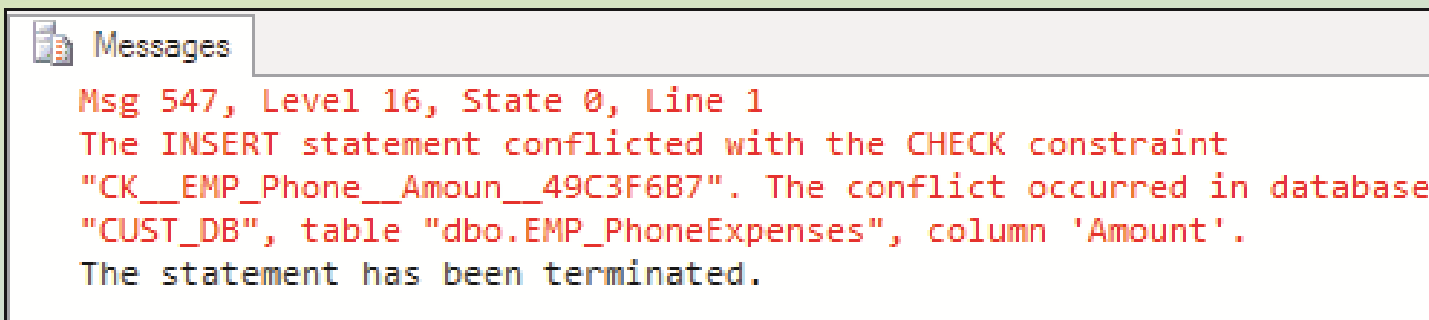
- Tuy nhiên vẫn có thể thêm giá trị NULL vào cột **Amount** nếu không biết giá trị của **Amount**.

## CHECK 2-2

- Khi có một ràng buộc CHECK được định nghĩa, nếu câu lệnh INSERT chèn dữ liệu vi phạm ràng buộc, nó sẽ nhận được lỗi như đoạn code dưới đây:

```
USE [CUST_DB]
INSERT INTO dbo.EMP_PhoneExpenses values (101, 983345674, 9)
GO
```

- Thông báo lỗi của đoạn code trên xuất hiện như hình dưới đây khi giá trị cho cột **Amount** nhỏ hơn **10**:



# NOT NULL

Ràng buộc NOT NULL được sử dụng để đảm bảo cột không nhận các giá trị NULL.

Ràng buộc NOT NULL là ràng buộc toàn vẹn về miền tương tự như ràng buộc CHECK.

- Một kiểu dữ liệu là một thuộc tính xác định kiểu dữ liệu mà một đối tượng có thể lưu trữ, chẳng hạn như là dữ liệu số, dữ liệu kí tự, dữ liệu tiền tệ, và như vv..
- SQL Server 2012 hỗ trợ ba loại kiểu dữ liệu khác nhau:
  - Kiểu dữ liệu hệ thống (System data types)
  - Kiểu dữ liệu bí danh (Alias data types)
  - Kiểu dữ liệu bí danh (User-defined types)
- Hầu hết các bảng đều có một khóa chính(primary key), được tạo ra từ một hoặc nhiều cột của bảng để xác định các bản ghi(record) là duy nhất.
- Đặc tính cho phép null của một cột là để xác định các dòng trong bảng có chứa giá trị null tại cột đó hay không.

- Một định nghĩa `DEFAULT` có thể được tạo một cột tại thời điểm tạo bảng hoặc được thêm vào sau khi bảng đã tồn tại.
- Thuộc tính `IDENTITY` của SQL Server được sử dụng để tạo ra các cột định danh, chúng chứa các giá trị tự động phát sinh tuần tự để nhận dạng duy nhất mỗi hàng trong một bảng.
- Một constraint là một thuộc tính được gắn tới cột để áp dụng các nguyên tắc xử lý logic (business logic rule) và đảm bảo tính toàn vẹn.
- Một ràng buộc `UNIQUE` được sử dụng để đảm bảo các giá trị được nhập vào một cột hoặc nhóm cột phải duy nhất (không trùng lặp).
- Khóa ngoại trong một bảng là một cột mà chỉ đến một khóa chính trong một bảng khác.
- Ràng buộc `CHECK` được sử dụng để giới hạn các giá trị có thể được nhập vào một cột.