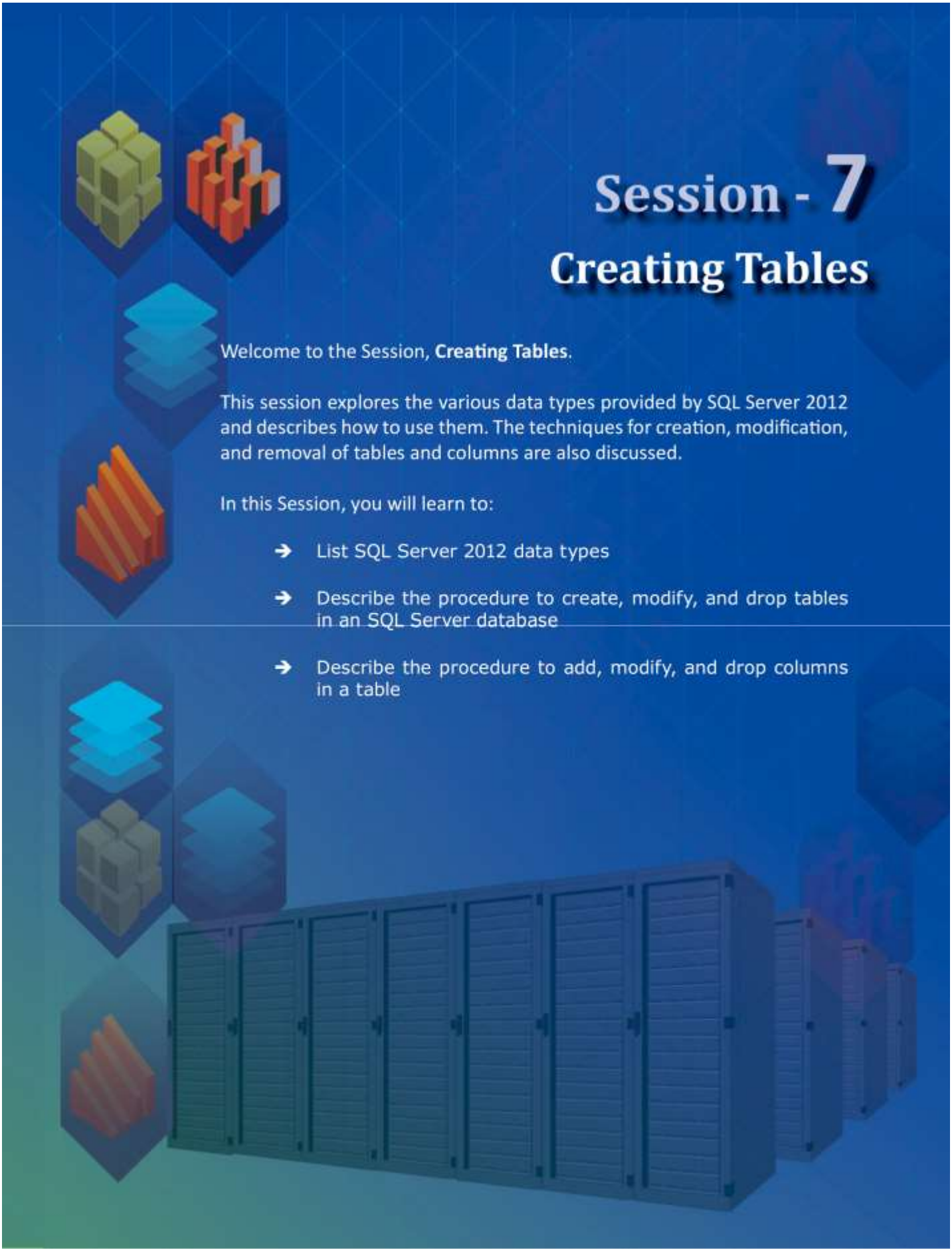


Session - 7

Creating Tables

Welcome to the Session, **Creating Tables**.

This session explores the various data types provided by SQL Server 2012 and describes how to use them. The techniques for creation, modification, and removal of tables and columns are also discussed.

In this Session, you will learn to:

- List SQL Server 2012 data types
- Describe the procedure to create, modify, and drop tables in an SQL Server database
- Describe the procedure to add, modify, and drop columns in a table

7.1 Giới thiệu

Một trong những loại đối tượng cơ sở dữ liệu quan trọng nhất trong SQL Server 2012 là bảng. Bảng trong SQL Server 2012 chứa dữ liệu dưới dạng các hàng và cột. Mỗi cột có thể có dữ liệu thuộc một loại và kích thước cụ thể.

7.2 Kiểu dữ liệu

Kiểu dữ liệu là một thuộc tính chỉ ra loại dữ liệu một đối tượng có thể giữ, như là dữ liệu số, dữ liệu ký tự, dữ liệu tiền tệ, và vân vân. Kiểu dữ liệu còn chỉ ra dung lượng lưu trữ của một đối tượng. Một khi cột đã được định nghĩa để lưu trữ dữ liệu thuộc về một kiểu dữ liệu đặc biệt, dữ liệu của loại khác không thể được lưu trữ trong đó. Theo cách này, các kiểu dữ liệu bắt buộc sự toàn vẹn dữ liệu

Xem xét một cột tên là **BasicSalary** trong bảng **Employee**. Để đảm bảo rằng chỉ có dữ liệu số được nhập vào, cột này được định nghĩa thuộc về một kiểu dữ liệu số nguyên. Do đó, nếu cố nhập dữ liệu ký tự vào cột BasicSalary đó, nó sẽ không thành công.

7.2.1 Các loại kiểu dữ liệu khác nhau

SQL Server 2012 hỗ trợ ba loại dữ liệu :

- **Các kiểu dữ liệu hệ thống –** Chúng được cung cấp bởi SQL Server 2012.
- **Các kiểu dữ liệu bí danh -** Chúng dựa trên các kiểu dữ liệu do hệ thống cung cấp. Một trong những cách sử dụng điển hình của các kiểu dữ liệu bí danh là khi nhiều hơn một bảng lưu trữ cùng một loại dữ liệu trong một cột và có đặc điểm tương tự như là chiều dài, khả năng nul , và loại. Trong các trường hợp như vậy, có thể tạo ra loại dữ liệu bí danh để có thể được dùng chung cho các bảng này.
- **Loại do người dùng định nghĩa** - Những loại này được tạo ra bằng cách sử dụng ngôn ngữ lập trình được .NET Framework hỗ trợ, đó là một khuôn khổ phần mềm được Microsoft phát triển.

Table 7.1 trình bày các kiểu dữ liệu khác nhau trong SQL Server 2012 cùng với thể loại và mô tả của chúng.

Loại	Loại dữ liệu	Mô tả
Các số chính xác	int	Trình bày một cột chiếm 4 byte không gian bộ nhớ. Thường được dùng để giữ các giá trị số nguyên.
	smallint	Trình bày một cột chiếm 2 byte không gian bộ nhớ. Có thể giữ dữ liệu số nguyên từ -32,768 to 32,767.
	tinyint	Trình bày một cột chiếm 1 byte không gian bộ nhớ. Có thể giữ dữ liệu số nguyên từ 0 to 255.

Loại	Loại Dữ liệu	Mô tả
Các số chính xác	bigint	Trình bày một cột chiếm 8 byte không gian bộ nhớ. Có thể giữ dữ liệu trong phạm vi -2^63 (-9.223.372.036.854.775.808) đến 2^63-1 (9.223.372.036.854.775.807)
	numeric	Trình bày một cột thuộc loại này cố định độ chính xác và quy mô
	money	Trình bày một cột chiếm 8 byte không gian bộ nhớ. Biểu diễn các giá trị dữ liệu tiền tệ dao động từ (–2^63/10000) (–922.337.203.685.477,5808) đến 2^63 –1(922.337.203.685.477,5807).
Các số gần đúng	float	Trình bày một cột chiếm 8 byte không gian bộ nhớ. Biểu diễn số có dấu thập phân thay đổi dao động từ –1.79E +308 đến 1.79E+308.
	real	Trình bày một cột chiếm 4 byte không gian bộ nhớ. Biểu diễn số có độ chính xác thay đổi dao động từ –3.40E+38 đến 3.40E+38.
Date và Time	datetime	Biểu diễn ngày và giờ. Được lưu như hai số nguyên 4 byte.
	smalldatetime	Biểu diễn ngày và giờ.
Chuỗi kí tự	char	Lưu trữ dữ liệu ký tự có chiều dài cố định và không phải là Unicode.
	varchar	Lưu dữ liệu ký tự có độ dài biến đổi và không phải Unicode.
	text	Lưu dữ liệu ký tự có độ dài biến đổi và không phải Unicode.
Loại Unicode	nchar	Lưu dữ liệu ký tự Unicode với độ dài cố định
	nvarchar	Lưu dữ liệu ký tự Unicode có độ dài biến đổi.

Concepts

Loại	Loại dữ liệu	Mô tả
Các loại dữ liệu khác	Timestamp	Trình bày một cột chiếm 8 byte không gian bộ nhớ. Có thể giữ tự động các số nhị phân duy nhất, được tạo ra mà được tạo ra cho một cơ sở dữ liệu.
	binary(n)	Lưu trữ dữ liệu nhị phân có chiều dài cố định với chiều dài tối đa là 8000 bytes.
Các loại dữ liệu khác	varbinary(n)	Lưu dữ liệu nhị phân có độ dài biến đổi với độ dài tối đa 8000 byte.
	image	Lưu dữ liệu nhị phân với độ dài biến đổi với độ dài tối đa là 2^30–1 (1.073.741.823) byte.
	uniqueidentifier	Trình bày một cột chiếm 16 byte không gian bộ nhớ. Ngoài ra, lưu trữ mã định danh duy nhất toàn cầu (GUID).

Table 7.1: Data Types in SQL Server 2012

Các loại dữ liệu bí danh có thể được tạo ra bằng cách sử dụng câu lệnh CREATE TYPE.

Cú pháp cho câu 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 sơ đồ trong đó loại dữ liệu bí danh đang được tạo ra. type_name: chỉ ra tên của loại bí danh đang được tạo ra.
base_type: chỉ ra tên của loại dữ liệu do hệ thống định nghĩa dựa trên đó loại dữ liệu bí danh đang được tạo ra.
precision và scale: chỉ ra độ chính xác và tỉ lệ cho dữ liệu số.
NULL|NOT NULL: chỉ ra xem loại dữ liệu có thể giữ giá trị null hay không.

Code Snippet 1 trình bày cách để tạo ra một loại dữ liệu bí danh có tên là **usertype** sử dụng câu lệnh CREATE TYPE.

Code Snippet 1:

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

Concepts

7.3 Tạo, sửa đổi, và thả bỏ các bảng

Hầu hết các bảng có một khóa chính, tạo thành từ một hoặc nhiều cột của bảng. Khóa chính luôn luôn là duy nhất. Database Engine sẽ thực thi hạn chế mà bất kỳ giá trị khóa chính nào không thể được lặp lại trong bảng. Do đó, khóa chính có thể được sử dụng để nhận dạng mỗi bản ghi một cách duy nhất.

7.3.1 Tạo bảng

Câu lệnh CREATE TABLE được sử dụng để tạo ra các bảng trong SQL Server 2012. Cú pháp cho câu 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 cơ sở dữ liệu trong đó bảng được tạo ra.
table_name: là tên của bảng mới.Snnbtable_nameSnnbcó thể có tối đa 128 ký tự. column_name: là tên của một cột trong bảng. column_name có thể lên đến 128 ký tự.
column_name : không được chỉ định cho các cột được tạo ra với kiểu dữ liệu timestamp . Tên cột mặc định của cột timestamp là timestamp.
data_type: Nó chỉ ra kiểu dữ liệu của cột.

Code Snippet 2 trình bày việc tạo ra một bảng có tên là **dbo.Customer_1**.

Code Snippet 2:

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

Một vài phần tiếp theo xem qua các đặc điểm khác nhau liên quan với các bảng như là tính chất không xác định của cột, định nghĩa mặc định, ràng buộc, và vân vân.

7.3.2 Sửa đổi bảng

Câu lệnh ALTER TABLE được sử dụng để sửa đổi một định nghĩa bảng bằng cách thay đổi, thêm, hoặc thả bỏ các cột và ràng buộc, gán lại các phân vùng, hoặc vô hiệu hóa hoặc cho phép các ràng buộc và bộ kích khởi.

Cú pháp cho câu lệnh ALTER TABLE như sau:

Cú pháp:

```
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 rằng cột cụ thể sẽ được thay đổi hoặc sửa đổi.

ADD : chỉ ra rằng một hoặc nhiều định nghĩa cột sẽ được thêm vào.

DROP COLUMN ([<column_name>] : chỉ ra rằng column_name sẽ được loại bỏ khỏi bảng.

Code Snippet 3 trình bày việc thay đổi cột **Customer_id**.

Code Snippet 3:

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

Code Snippet 4 trình bày việc thêm cột **Contact_number**.

Code Snippet 4:

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

Code Snippet 5 trình bày việc thả rơi cột **Contact_number.**

Code Snippet 5:

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

Tuy nhiên, trước khi thử xóa các cột, quan trọng là bảo đảm các cột có thể xóa được. Trong điều kiện nhất định, không thể thả rơi các cột, chẳng hạn như khi chúng được sử dụng trong ràng buộc CHECK, FOREIGN KEY, UNIQUE, hoặc PRIMARY KEY, kết hợp với định nghĩa DEFAULT, và ..vv

7.3.3 Thả rơi các bảng

Câu lệnh **DROP TABLE** loại bỏ một định nghĩa bảng, dữ liệu của nó, và tất cả các đối tượng liên quan như là các chỉ số, trigger, ràng buộc, và thông số kỹ thuật cho phép đối với bảng đó.
Cú pháp cho câu lệnh DROP TABLE như sau:

Cú pháp:

```
DROP TABLE <Table_Name>
```

trong đó,

<Table_Name> : là tên của bảng sẽ được thả rơi.

Code Snippet 6 trình bày cách để thả rơi một bảng.

Code Snippet 6:

```
USE [CUST_DB]

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

7.3.4 Các câu lệnh sửa đổi dữ liệu

Câu lệnh được sử dụng để sửa đổi dữ liệu là các câu lệnh INSERT, UPDATE, và DELETE. Điều này được giải thích như sau:

- **Câu lệnh INSERT** - Câu lệnh INSERT thêm một hàng mới vào bảng. Cú pháp cho câu lệnh INSERT như sau:

  Cú pháp:

  ```
  INSERT [INTO] <Table_Name>

  VALUES <values>
  ```

  trong đó,

  <Table_Name> : là tên của bảng trong đó hàng sẽ được chèn vào.

  [INTO] : là một từ khóa tùy chọn được sử dụng giữa INSERT và bảng mục tiêu.

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

Concepts

Code Snippet 7 trình bày việc thêm một hàng mới vào bảng **Table_2. Code**

Snippet 7:

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

Kết quả của điều này sẽ là một hàng với dữ liệu đã cho được đưa vào bảng.

- **Câu lệnh UPDATE** - Câu lệnh UPDATE sửa đổi dữ liệu trong bảng. Cú pháp cho câu lệnh UPDATE như sau:

Cú pháp:

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

trong đó,

<Table_Name>: là tên của bảng trong đó các bản ghi sẽ được cập nhật.

<Column_Name>: là tên của cột trong bảng trong đó bản ghi sẽ được cập nhật.

<Value>: chỉ ra giá trị mới cho cột đã sửa đổi.

<Search condition>: chỉ ra điều kiện được đáp ứng cho các hàng bị xóa.

Code Snippet 8 trình bày việc sử dụng câu lệnh **UPDATE** để sửa đổi giá trị trong cột **Contact_number.**

Code Snippet 8:

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

Hình 7.1shows the output of UPDATE statement.

Results | Messages

	Customer_id number	Customer_name	Contact_name	Contact_number
1	101	Richard Parker	Richy	5432679

Hình 7.1: **Kết quả của câu lệnh UPDATE**

Concepts

- **Câu lệnh DELETE** - Câu lệnh DELETE loại bỏ các hàng khỏi bảng. Cú pháp cho câu lệnh DELETE như sau :

Cú pháp :

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

trong đó,

<Table_Name>: là tên của bảng mà từ đó các bản ghi sẽ được xóa.

Mệnh đề WHERE được sử dụng để chỉ ra tình trạng này. Nếu mệnh đề WHERE không được đưa vào trong câu lệnh DELETE, tất cả các bản ghi trong bảng sẽ bị xóa.

Code Snippet 9 trình bày cách xóa một hàng khỏi bảng **Customer_2** nơi giá trị **Contact_number** bằng 5.432.679.

Code Snippet 9:

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

7.3.5 Cột khả năng Null

Đặc điểm tính chất không xác định của một cột xác định xem các hàng trong bảng có thể chứa giá trị null cho cột đó hay không. Trong SQL Server, giá trị null không giống như bằng không, trống, hoặc một chuỗi ký tự có độ dài bằng không (chẳng hạn như ' '). Ví dụ, giá trị null trong cột Color của bảng Production. Product của cơ sở dữ liệu AdventureWorks2012 không có nghĩa là sản phẩm không có màu, nó chỉ có nghĩa là màu của sản phẩm là không rõ hoặc chưa được đặt.

Có thể định nghĩa tính chất không xác định khi tạo ra bảng hoặc khi sửa đổi bảng.

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

Khi chèn một hàng, nếu không có giá trị được đưa ra cho một cột nullable (có nghĩa là, nó cho phép các giá trị null), khi đó, SQL Server tự động cung cấp cho nó một giá trị null trừ khi cột đã được đưa ra một định nghĩa mặc định. Cũng có thể nhập riêng một giá trị null vào trong một cột bất kể nó thuộc loại dữ liệu nào hoặc là nó có giá trị mặc định nào được liên kết với nó. Việc làm cho một cột thành không thể không xác định (đó là không cho phép các giá trị null) làm tăng cường tính toàn vẹn dữ liệu bằng cách bảo đảm là cột có chứa dữ liệu trong mọi hàng.

Trong Code Snippet 10, câu lệnh CREATE TABLE sử dụng từ khóa NULL và NOT NULL với định nghĩa cột.

Code Snippet 10:

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

Kết quả của mã là bảng **StoreDetails** được tạo ra với cột **StoreID** và **Name** bổ sung vào bảng này**.**

7.3.6 Định nghĩa DEFAULT

Xem xét kịch bản trong đó các chi tiết về sản phẩm cần được lưu trong bảng của SQL Server 2012 nhưng tất cả các giá trị cho chi tiết sản phẩm có thể không rõ thậm chí vào thời điểm chèn vào dữ liệu. Tuy nhiên, theo các quy tắc tính toàn vẹn và sự nhất quán dữ liệu, các cột trong một bản ghi thường chứa một giá trị. Việc lưu các giá trị null vào trong các cột như vậy khi giá trị chính xác của dữ liệu không rõ có thể là không thích hợp hoặc không thực tế.

Trong các tình huống như vậy, định nghĩa DEFAULT có thể được đưa ra cho cột để gán nó làm giá trị mặc định nếu không có giá trị nào được đưa ra vào thời điểm tạo ra. Ví dụ, người ta thường chỉ ra số không làm giá trị mặc định cho các cột số hoặc 'N/A' hoặc 'Unknown' làm giá trị mặc định cho các cột chuỗi khi không có giá trị nào được chỉ ra.

Định nghĩa DEFAULT cho một cột có thể được tạo ra vào thời điểm tạo bảng hoặc thêm ở giai đoạn sau vào bảng hiện có. Khi định nghĩa DEFAULT được thêm vào một cột hiện có trong bảng, SQL Server áp dụng các giá trị mặc định mới chỉ cho những dòng dữ liệu đó, đã mới được thêm vào bảng.

Trong Code Snippet 11 , câu lệnh CREATE TABLE sử dụng từ khóa DEFAULT để định nghĩa giá trị mặc định cho Price.

Code Snippet 11:

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

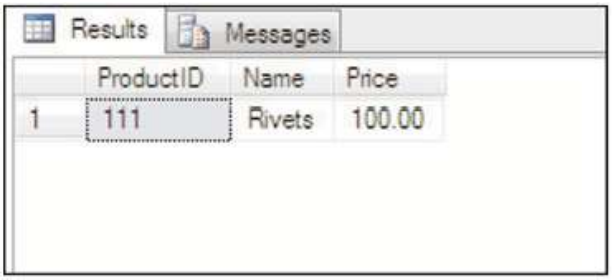
Khi một hàng được đưa vào sử dụng câu lệnh như trong Đoạn mã 12, giá trị của **Price** sẽ không được để trống, nó sẽ có giá trị 100.00 mặc dù người dùng đã không nhập bất kỳ giá trị nào cho cột đó.

Code Snippet 12:

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

Concepts

Hình 7.2 trình bày kết quả của Code Snippet 12nơi mặc dù các giá trị được thêm vào chỉ cho cột **ProductID** và **Name**, cột **Price** vẫn sẽ hiển thị giá trị 100.00. Điều này là do định nghĩa DEFAULT.



Hình 7.2:Values added to ProductiD and Name Columns

Không thể tạo ra phần sau đây trên các cột với định nghĩa DEFAULT:

- Loại dữ liệu timestamp
- Thuộc tính IDENTITY hoặc ROWGUIDCOL
- Một định nghĩa mặc định hiện hữu hoặc đối tượng mặc định

7.3.7 Thuộc tính IDENTITY

Thuộc tính IDENTITY của SQL Server được sử dụng để tạo ra các cột mã định danh có thể chứa các giá trị liên tục tự động tạo ra để xác định duy nhất mỗi hàng trong một bảng. Ví dụ, một cột định danh có thể được tạo ra để tạo ra các số đăng ký học viên duy nhất một cách tự động bất cứ khi nào các hàng mới được đưa vào bảng Students. Số nhận dạng cho hàng đầu tiên được chèn vào bảng được gọi là giá trị hạt giống (seed) và lượng tăng, còn được gọi là thuộc tính Lượng tăng đồng nhất, được thêm vào hạt giống để tạo ra thêm các số nhận dạng theo tuần tự. Khi một hàng mới được chèn vào một bảng có một cột nhận dạng, giá trị nhận dạng tiếp theo được SQL Server tạo ra tự động bằng cách thêm lượng tăng vào hạt giống. Cột nhận dạng thường được dùng cho các giá trị khóa chính.

Các đặc tính của thuộc tính IDENTITY như sau:

- Một cột có thuộc tính IDENTITY phải được định nghĩa bằng cách sử dụng một trong những kiểu dữ liệu sau: decimal, int, numeric, smallint, bigint, hoặc tinyint.
- Một cột có thuộc tính IDENTITY không cần phải có mầm và giá trị tăng dần được chỉ ra. Nếu chúng không được chỉ ra, giá trị mặc định bằng 1 sẽ được dùng cho cả hai.
- Một bảng không thể có nhiều hơn một cột với thuộc tính IDENTITY.
- Cột mã định danh trong một bảng phải không cho phép giá trị null và không được chứa định nghĩa hoặc đối tượng DEFAULT.

Concepts

- Cột được định nghĩa với thuộc tính IDENTITY không thể có các giá trị của chúng được cập nhật.

- Những giá trị này có thể được chèn vào một cách rõ ràng vào cột nhận dạng của một bảng chỉ khi tùy chọn IDENTITY _ INSERT được đặt là ON. Khi IDENTITY _ INSERT là ON, các câu lệnh INSERT phải cung cấp một giá trị.

Ưu điểm của các cột nhận dạng là SQL Server có thể các cột tự động các giá trị khóa, như vậy sẽ làm giảm chi phí và cải thiện hiệu suất. Việc sử dụng các cột nhận dạng làm đơn giản hóa việc lập trình và giữ cho các giá trị khóa chính ngắn gọn.

Một khi thuộc tính IDENTITY đã được đặt, lấy giá trị của cột mã định danh có thể được thực hiện bằng cách sử dụng từ khóa IDENTITYCOL với tên bảng trong câu lệnh SELECT. Để biết xem một bảng có một cột IDENTITY hay không, hàm OBJECTPROPERTY() có thể được sử dụng. Để lấy ra tên của cột IDENTITY trong một bảng, hàm COLUMNPROPERTY được sử dụng.

Cú pháp cho thuộc tính IDENTITY 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 mầm từ đó để bắt đầu tạo ra giá trị nhận dạng.
increment_value: là giá trị tăng dần theo đó để tăng mỗi lần.

Code Snippet 13 trình bày việc sử dụng thuộc tính IDENTITY. **HRContactPhone** được tạo ra như một bảng với hai cột trong lược đồ **Person** có sẵn trong cơ sở dữ liệu **CUST_DB**. Cột **Person_ID** là một cột nhận dạng. Giá trị hạt giống là 500, và giá trị gia tăng là 1.

Code Snippet 13:

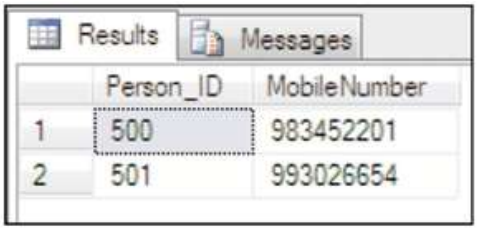
```
USE [CUST_DB]
GO
CREATE TABLE HRContactPhone ( Person_ID int IDENTITY(500,1) NOT NULL,
MobileNumber bigint NOT NULL )
GO
```

Trong khi chèn các hàng vào bảng, nếu IDENTITY_INSERT không được bật, khi đó, không thể cho các giá trị rõ ràng cho cột IDENTITY. Thay vào đó, có thể cho các câu lệnh tương tự như Code Snippet 14

Code Snippet 14:

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

Hình 7.3 trình bày kết quả trong đó thuộc tính IDENTITY làm tăng các giá trị cột **Person_ID**.



Hình 7.3: IDENTITY Property Applied on Person_ID Column

7.3.8 Mã định danh duy nhất toàn cầu

Ngoài thuộc tính IDENTITY, SQL Server còn hỗ trợ các mã định danh duy nhất toàn cầu. Thông thường, trong một môi trường được nối mạng, nhiều bảng có thể cần có một cột chứa giá trị đơn nhất tổng thể chung. Xem xét kịch bản khi dữ liệu từ nhiều hệ cơ sở dữ liệu như các cơ sở dữ liệu ngân hàng phải được hợp nhất ở chỉ một vị trí. Khi dữ liệu từ khắp thế giới được đối chiếu ở địa điểm trung tâm về sự hợp nhất và lập báo cáo, sử dụng các giá trị đơn nhất tổng thể để tránh khách hàng ở các quốc gia khác nhau không có cùng số tài khoản ngân hàng hoặc mã ID khách hàng. Để thỏa mãn nhu cầu này, SQL Server cung cấp các cột mã nhận dạng đơn nhất tổng thể. Có thể tạo ra các cột này cho từng bảng chứa các giá trị đơn nhất trên tất cả các máy tính trong một mạng. Chỉ có thể tạo ra một cột mã nhận dạng và một cột mã nhận dạng đơn nhất tổng thể cho mỗi bảng. Để tạo và làm việc với các mã định danh duy nhất toàn cầu, một sự kết hợp của ROWGUIDCOL, kiểu dữ liệu uniqueidentifier, và hàm NEWID được sử dụng.

Các giá trị cho cột đơn nhất tổng thể không được tạo ra tự động. Người ta phải tạo ra một định nghĩa DEFAULT với hàm NEWID() cho cột uniqueidentifier để tạo ra một giá trị duy nhất toàn cầu. Hàm NEWID() tạo ra một số định danh duy nhất là một chuỗi nhị phân 16-byte. Cột có thể được tham chiếu trong danh sách SELECT bằng cách sử dụng từ khóa ROWGUIDCOL.

Để biết xem một bảng có cột ROWGUIDCOL hay không, hàm OBJECTPROPERTY được sử dụng. Hàm COLUMNPROPERTY được sử dụng để lấy tên của cột ROWGUIDCOL. Code Snippet 15 trình bày cách câu lệnh CREATE TABLE tạo ra bảng **EMPCellularPhone**.

Cột **Person_ID** tự động tạo ra một GUID cho mỗi hàng mới được thêm vào bảng.

Code Snippet 15:

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

Code Snippet 16 thêm một giá trị cho cột **PersonName**.

Code Snippet 16:

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

Hình 7.4 trình bày kết quả trong đó mã định danh duy nhất được hiển thị với một **PersonName** cụ thể.

Hình 7.4: Unique Identifier

7.4 Các ràng buộc

Một trong các chức năng quan trọng của SQL Server là duy trì và tăng cường tính toàn vẹn dữ liệu. Có một số phương tiện để đạt được điều này nhưng một trong các phương pháp được dùng phổ biến và được ưa chuộng là sử dụng các ràng buộc. Ràng buộc là một thuộc tính được gán cho một cột hoặc tập hợp cột trong một bảng để ngăn một số loại giá trị dữ liệu không nhất quán không được nhập vào. Các ràng buộc được dùng để áp dụng các quy tắc logic trong kinh doanh và tăng cường tính toàn vẹn dữ liệu.

Các ràng buộc có thể được tạo ra khi bảng được tạo ra, như là một phần của định nghĩa bảng bằng cách sử dụng câu lệnh CREATE TABLE hoặc có thể được thêm vào ở giai đoạn sau bằng cách sử dụng câu lệnh ALTER TABLE. Có thể phân loại các ràng buộc thành ràng buộc cột và ràng buộc bảng. Ràng buộc cột được chỉ ra như một phần của định nghĩa cột và chỉ áp dụng cho cột đó. Ràng buộc bảng có thể áp dụng cho nhiều hơn một cột trong bảng và được khai báo độc lập khỏi định nghĩa cột. Ràng buộc bảng phải được dùng khi có nhiều hơn một cột được đưa vào trong một ràng buộc.

SQL Server hỗ trợ các loại ràng buộc sau:

- **PRIMARY KEY**
- **UNIQUE**

Concepts

- **FOREIGN KEY**
- **CHECK**
- **NOT NULL**

7.4.1 PRIMARY KEY

Bảng thông thường có một khóa chính bao gồm một cột đơn hoặc tổ hợp các cột để nhận biết duy nhất từng hàng ở trong bảng. Ràng buộc PRIMARY KEY được sử dụng để tạo ra một khóa chính và thực thi tính toàn vẹn của thực thể của bảng. Chỉ có thể tạo ra một ràng buộc khóa chính cho một bảng. Hai hàng trong một bảng không thể có cùng một giá trị khóa chính và một cột là khóa chính không thể có các giá trị NULL. Do đó, khi ràng buộc khóa chính được thêm vào các cột hiện hữu của một bảng, SQL Server 2012 sẽ kiểm tra xem các quy tắc cho khóa chính có được tuân thủ hay không. Nếu dữ liệu hiện có trong các cột không tuân thủ các quy tắc cho khóa chính, khi đó ràng buộc sẽ không được thêm vào.

Cú pháp để thêm một khóa chính trong khi tạo ra một bảng như sau:
Cú pháp:

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

Code Snippet 17 trình bày cách tạo ra bảng **EMPContactPhone** để lưu trữ các chi tiết điện thoại liên lạc của một người. Do cột **EMP_ID** phải là khóa chính để xác định mỗi hàng duy nhất, nó được tạo ra với ràng buộc khóa chính.

Code Snippet 17:

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

Một phương pháp khác là sử dụng từ khóa CONSTRAINT. Cú pháp như sau:
Cú pháp:

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

Sau khi đã tạo ra một khóa chính cho **EMP_ID**, truy vấn được viết để chèn các hàng vào bảng với các câu lệnh được trình bày trong Code Snippet 18.

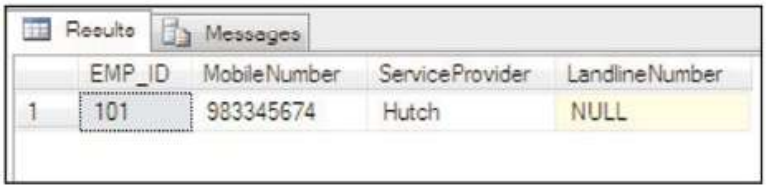
Code Snippet 18:

```
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 được trình bày trong Đoạn mã 18 được thực hiện thành công nhưng câu lệnh INSERT tiếp theo sẽ thất bại vì giá trị cho **EMP_ID** là trùng lặp như được trình bày trong hình 7.5.

Hình 7.5: Output Error Message for Duplicate EMP_ID

Kết quả của Code Snippet 18 được trình bầy trong Hình 7.6.



Hình 7.6: Output of the Executed First Statement

7.4.2 UNIQUE

Ràng buộc UNIQUE được sử dụng để đảm bảo rằng chỉ những giá trị duy nhất được nhập vào trong một cột hay tập hợp các cột. Nó cho phép người phát triển bảo đảm là không có giá trị trùng lặp được nhập vào. Các khóa chính là hoàn toàn duy nhất. Các ràng buộc khóa duy nhất bắt buộc toàn vẹn thực thể bởi vì một khi những ràng buộc này được áp dụng, không có hai hàng trong bảng có thể có cùng một giá trị cho những cột đó. Ràng buộc UNIQUE cho phép các giá trị null.

Bảng đơn lẻ có thể có nhiều hơn một ràng buộc UNIQUE.
Cú pháp để tạo ra ràng buộc UNIQUE như sau :
Cú pháp:

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

Concepts

Code Snippet 19 trình bày cách làm cho các cột **MobileNumber** và **LandlineNumber** là duy nhất.

Code Snippet 19:

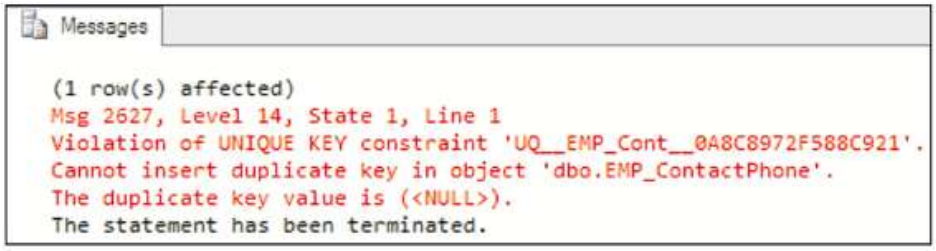
```
USE [CUST_DB]
GO
CREATE TABLE EMP_ContactPhone (Person_ID int PRIMARY KEY, MobileNumber bigint
UNIQUE, ServiceProvider varchar(30), LandlineNumber bigint UNIQUE)
```

Code Snippet 20 trình bày cách chèn một hàng vào bảng.

Code Snippet 20:

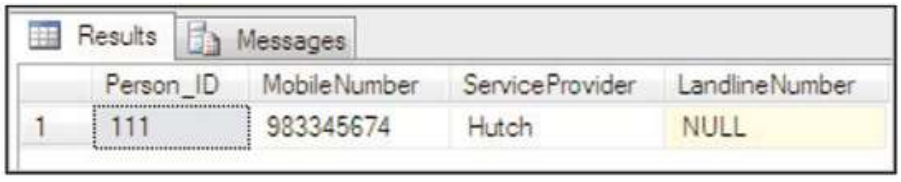
```
USE [CUST_DB]
INSERT INTO EMP_ContactPhone values (111, 983345674, 'Hutch', NULL)
INSERT INTO EMP_ContactPhone values (112, 983345674, 'Airtel', NULL)
GO
```

Mặc dù giá trị NULL đã được đưa ra cho các cột LandlineNumber, được định nghĩa là UNIQUE, lệnh sẽ thực hiện thành công bởi vì các ràng buộc UNIQUE chỉ kiểm tra tính duy nhất của các giá trị nhưng không ngăn chặn các mục nhập null. Câu lệnh đầu tiên được trình bày trong Code Snippet 20 được thực hiện thành công nhưng câu lệnh INSERT tiếp theo sẽ thất bại thậm chí giá trị khóa chính có khác vì giá trị cho **MobileNumber** là trùng lặp như được trình bày trong hình 7.7. Điều này là do cột MobileNumber được định nghĩa là duy nhất và không cho phép các giá trị trùng lặp.



Hình 7.7: Output Error Message for Value Duplicate MobileNumber

Kết quả của Code Snippet 20 được trình bầy trong Hình 7.8.



Hình 7.8: Output ofthe Executed UNIQUE Constraint

Concepts

7.4.3 FOREIGN KEY

Khóa ngoại trong bảng là một cột trỏ tới cột khóa chính trong bảng khác. Các ràng buộc khóa ngoại được dùng để tăng cường tính toàn vẹn tham chiếu. Cú pháp cho khóa ngoại như sau:

Cú pháp:

```
CREATE TABLE <table_name1>([column_list,]<column_name><datatype> FOREIGN KEY
REFERENCES <table_name> (pk_column_name> [, column_list])
```

trong đó,

table_name: là tên của bảng nơi tham khảo khóa chính từ đó.

<pk_column_name>: là tên của cột khóa chính.

Code Snippet 21 trình bày cách để tạo ra ràng buộc khóa ngoại.

Code Snippet 21:

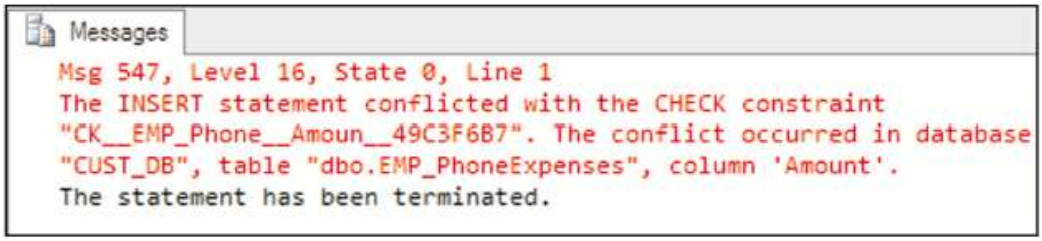
```
USE [CUST_DB]
GO
CREATE TABLE EMP_PhoneExpenses ( Expense_ID int PRIMARY KEY, MobileNumber bigint
FOREIGN KEY REFERENCES EMP_ContactPhone (MobileNumber), Amount bigint)
```

Một hàng được chèn vào bảng để số điện thoại di động cũng giống như một trong những số điện thoại di động trong **EMP_ContactPhone**. Lệnh này sẽ được viết được trình bày trong Code Snippet 22.

Code Snippet 22:

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

Thông báo lỗi của Code Snippet 22 được trình bầy trong Hình 7.9.



Hình 79: Output Error Message of FOREIGN KEY REFERENCES

Nếu không có khóa trong bảng tham chiếu có giá trị đang được đưa vào khóa ngoại, việc chèn vào sẽ thất bại như được trình bày trong hình 7.9. Tuy nhiên, có thể thêm giá trị NULL vào một cột khóa ngoại.

7.4.4 CHECK

Ràng buộc CHECK hạn chế các giá trị có thể được đặt trong một cột. Các ràng buộc kiểm tra tăng cường tính toàn vẹn dữ liệu. Ví dụ, ràng buộc CHECK có thể được đưa ra để kiểm tra xem giá trị đang được nhập vào **VoterAge** có lớn hơn hoặc bằng 18 hay không. Nếu dữ liệu đang được nhập cho cột không thoả mãn điều kiện này, khi đó, việc chèn vào sẽ thất bại.

Ràng buộc CHECK hoạt động bằng cách chỉ ra một điều kiện tìm kiếm, có thể đánh giá thành TRUE, FALSE, hoặc không rõ. Các giá trị đánh giá thành FALSE bị từ chối. Nhiều ràng buộc CHECK có thể được chỉ ra cho một cột đơn lẻ. Ràng buộc CHECK đơn lẻ cũng có thể được áp dụng cho nhiều cột bằng cách tạo ra nó ở mức bảng.

Code Snippet 23 trình bày việc tạo ra một ràng buộc CHECK để đảm bảo rằng giá trị **Amount** sẽ luôn luôn khác không. Tuy nhiên, giá trị NULL có thể được thêm vào cột **Amount** nếu giá trị của **Amount** là không biết

Code Snippet 23:

```
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
```

Một khi ràng buộc CHECK đã được định nghĩa, nếu câu lệnh INSERT được viết với dữ liệu vi phạm ràng buộc này, nó sẽ thất bại như được trình bày trong Code Snippet 24.

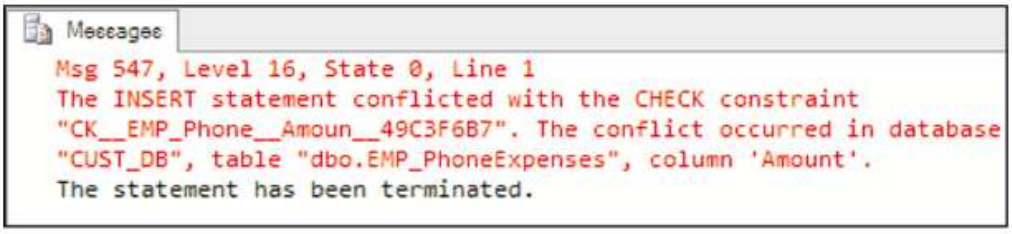
Code Snippet 24:

```
USE [CUST_DB]

INSERT INTO dbo.EMP_PhoneExpenses values (101, 983345674, 9)

GO
```

Thông báo lỗi của Code Snippet 24 xuất hiện khi ràng buộc **Amount** nhỏ hơn 10 được trình bày trong Hình 7.10.



Hình 7.10: Output Error Message of CHECK Constraint

7.4.5 NOT NULL

Ràng buộc NOT NULL bắt buộc cột sẽ không chấp nhận các giá trị null. Các ràng buộc NOT NULL được sử dụng để bắt buộc tính toàn vẹn miền, tương tự như ràng buộc CHECK.

7.5 Kiểm tra tiến độ của bạn

1. Tính năng nào sau đây của cột xác định xem các hàng trong bảng có thể chứa một giá trị null cho cột đó?

(A)	Default	(C)	Group BY
(B)	Multiplicity	(D)	Nullability

2. Một_____trong một bảng là một cột trỏ tới một cột khóa chính trong bảng khác.

(A)	Foreign key	(C)	Repeated key
(B)	Secondary key	(D)	Local key

3. Mã nào sau đây được sử dụng để thả một bảng từ cơ sở dữ liệu **CUST _ DB**?

(A)	DROP TABLE [dbo].[Table_1]	(C)	USE [CUST_DB] GO DELETE TABLE [dbo].[Table_1]
(B)	USE [CUST_DB] GO DROP TABLE [dbo].[Table_1]	(D)	USE [CUST_DB] GO SUBTRACT [dbo].[Table_1]

4. Thuộc tính nào sau đây của SQL Server được sử dụng để tạo ra các cột mã định danh có thể chứa các giá trị liên tục tự động tạo ra để xác định duy nhất mỗi hàng trong một bảng ?

(A)	SELECT	(C)	INSERT
(B)	IDENTITY	(D)	DEFAULT

5. Ràng buộc_____được sử dụng để đảm bảo rằng chỉ những giá trị duy nhất được nhập vào trong một cột hay tập hợp các cột.

(A)	UNIQUE	(C)	Foreign key
(B)	DEFAULT	(D)	INSERT

Concepts

7.5.1 Đáp án

1.	D
2.	A
3.	B
4.	B
5.	A

Tóm tắt

- Kiểu dữ liệu là một thuộc tính chỉ ra dung lượng lưu trữ của đối tượng và loại dữ liệu nó có thể giữ, như là dữ liệu số, dữ liệu ký tự, dữ liệu tiền tệ, và ..vv.
- SQL Server 2012 hỗ trợ ba loại dữ liệu:
  - Các kiểu dữ liệu hệ thống
  - Loại dữ liệu alias
  - Các loại do người dùng định nghĩa
- Hầu hết các bảng có một khóa chính, được tạo thành từ một hoặc nhiều cột của bảng chỉ ra các bản ghi là duy nhất.
- Đặc điểm tính chất không xác định của một cột xác định xem các hàng trong bảng có thể chứa giá trị null cho cột đó hay không.
- Định nghĩa DEFAULT cho một cột có thể được tạo ra vào thời điểm tạo bảng hoặc thêm ở giai đoạn sau vào bảng hiện có.
- Thuộc tính IDENTITY của SQL Server được sử dụng để tạo ra các cột mã định danh có thể chứa các giá trị liên tục tự động tạo ra để xác định duy nhất mỗi hàng trong một bảng.
- Các ràng buộc được dùng để áp dụng các quy tắc logic trong kinh doanh và tăng cường tính toàn vẹn dữ liệu.
- Ràng buộc UNIQUE được dùng để bảo đảm là chỉ các giá trị duy nhất được nhập vào một cột hoặc tập hợp cột.
- Khóa ngoại trong bảng là một cột trỏ tới cột khóa chính trong bảng khác.
- Ràng buộc CHECK hạn chế các giá trị có thể được đặt vào một cột.

Concepts

Try It Yourself

1. Saint Clara Insurance (SCI) là một công ty bảo hiểm hàng đầu có trụ sở tại New York, Mỹ. SCI Services muốn có một cách nhanh hơn, chính xác hơn, và ít tốn kém hơn để xử lý các yêu cầu bảo hiểm điều chỉnh cho các khách hàng công ty bảo hiểm của họ. Với một cơ số khách hàng ngày càng tăng, họ quyết định tạo ra một ứng dụng dựa trên web sẽ được sử dụng không chỉ bởi các nhân viên làm việc tại hiện trường, nhưng cũng sẽ được sử dụng bởi các quản trị viên tại trụ sở chính.

   SCI xử lý khoảng 650 yêu cầu bảo hiểm mỗi tháng, nhưng có thể tăng lên đến 15.000 hoặc nhiều hơn khi có một cơn bão hoặc một số thảm họa khác. Các viên chức có thể sử dụng phần mềm trên loại thiết bị mà họ lựa chọn: Máy tính bảng hoặc máy tính xách tay ngoài hiện trường, hoặc máy tính để bàn tại văn phòng của họ. Việc sử dụng Microsoft SQL Server 2005 như làm cơ sở dữ liệu của phần mềm cho phép tiếp nhận và cập nhật tất cả các thông tin cần thiết liên quan đến khách hàng hoặc người yêu cầu bảo hiểm.

   Với hàng ngàn khách hàng dự kiến mỗi tháng, tính toàn vẹn dữ liệu của dữ liệu trong cơ sở dữ liệu là rất quan trọng. Bạn cần phải thực hiện các nhiệm vụ sau:

   a. Tạo ra một cơ sở dữ liệu được gọi là **SaintClaraServices** để lưu trữ các chi tiết của công ty. Tạo ra bảng **CustomerHeader** với các chi tiết được đưa ra trong table 7.2.

Tên trường	Loại dữ liệu	Mô tả
ClientiD	int	Mã khách hàng. Đây là Primary Key
FirstName	char	Tên khách hàng
LastName	char	Họ khách hàng
MiddleName	char	Tên đệm của khách hàng
Gender	char	Giới tính
DateOfBirth	datetime	Ngày sinh
Address	varchar(max)	Địa chỉ của khách hàng
MaritaiStatus	char	Trạng thái hôn nhân của khách hàng
Age	int	Tuổi của khách hàng
Employment	char	Nghề nghiệp của khách hàng
CompanyName	varchar(max)	Tên công ty
CompanyAddress	varchar(max)	Địa chỉ công ty

Table 7.2: CustomerHeader Table

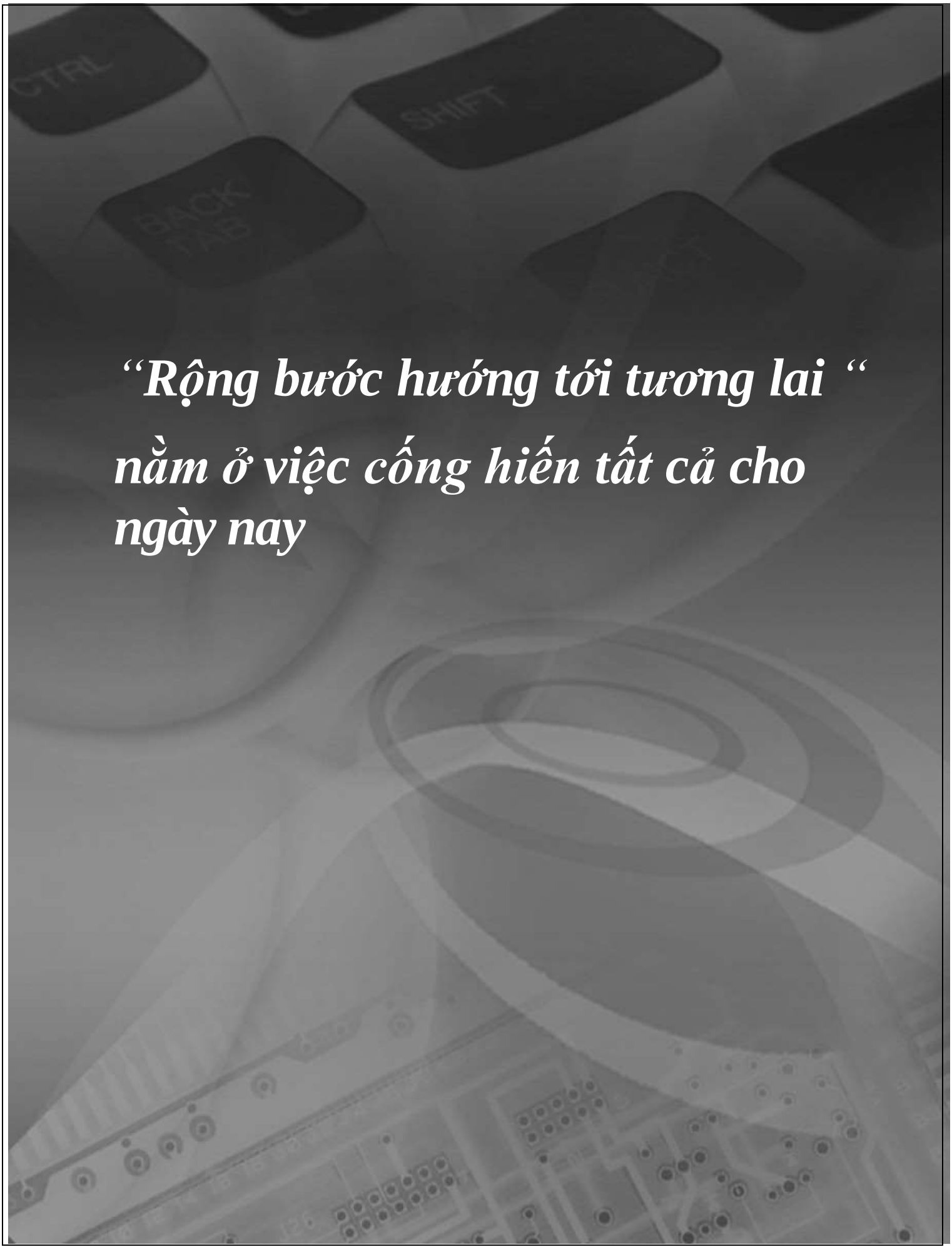
Try It Yourself

b. Tạo ra bảng **CustomerDetails** với các thông số kỹ thuật được đưa ra trong table 7.3.

Tên trường	Loại dữ liệu	Mô tả
ClientiD	int	Mã Khách hàng. Đây là Primary Key
FatherName	char	Tên của cha khách hàng
MotherName	char	Tên của mẹ khách hàng
Amount	money	Lưu trữ tiền vốn vay
Period	int	Giai đoạn bảo hiểm
Plan	char	Chương trình bảo hiểm
Premium	money	Phí bảo hiểm
NomineeName	char	Tên người đứng tên
Date	datetime	Lưu trữ ngày bảo hiểm được thực hiện

Table 7.3: CustomerDetails Table

c. Thêm một foreign key vào table **CustomerDetails** .



“Rộng bước hướng tới tương lai “ nằm ở việc cống hiến tất cả cho ngày nay