BACHKHOA EDUCATION APTECH
Làm trước học sau

Aptech COMPUTER EDUCATION

TRƯỜNG ĐÀO TẠO LẬP TRÌNH VIÊN VÀ QUẢN TRỊ MẠNG QUỐC TẾ BACHKHOA-APTECH

Bài 07
Tạo bảng

Tóm tắt

- Các lệnh tạo, sửa, xóa bảng
- Các lệnh thêm, sửa, xóa dữ liệu trong bảng
- Các ràng buộc trên cột
- Một số thuộc tính/định nghĩa trên cột

Table of baby-name data
(baby-2010.csv)

name	rank	gender	year
Jacob	1	boy	2010
Isabella	1	girl	2010
Ethan	2	boy	2010
Sophia	2	girl	2010
Michael	3	boy	2010

Field names

One row (4 fields)

2000 rows all told

Bảng

Lệnh **CREATE TABLE** sử dụng để tạo bảng

Cú pháp

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

Bảng

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.

Bảng

Ví dụ:

```
-- Tạo bảng LOAI_SAN_PHAM
CREATE TABLE LOAI_SAN_PHAM (
ma_lsp int identity(1,1) PRIMARY KEY, -- Khóa chính, kiểu int, tự tăng
ten_loai nvarchar(50) NOT NULL, -- Không được để trống
trang_thai tinyint DEFAULT(1) -- Mặc định dữ liệu là 1
)
GO
```

Bảng

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:

```
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>];
```

Bảng

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.

Bảng

Sửa cột:

```
-- Sửa kiểu dữ liệu cột giá nhập thành money
ALTER TABLE SAN_PHAM
ALTER COLUMN gia_nhap money
```

Thêm cột:

```
-- Thêm cột don_vi_tinh
ALTER TABLE SAN_PHAM
ADD don_vi_tinh nvarchar(16)
GO
```

Bảng

Xóa cột:

```
-- Xóa cột don_vi_tinh
ALTER TABLE SAN_PHAM
DROP don_vi_tinh
GO
```

Lưu ý: 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...

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

```
DROP TABLE <Table_Name>
```

Trong đó:

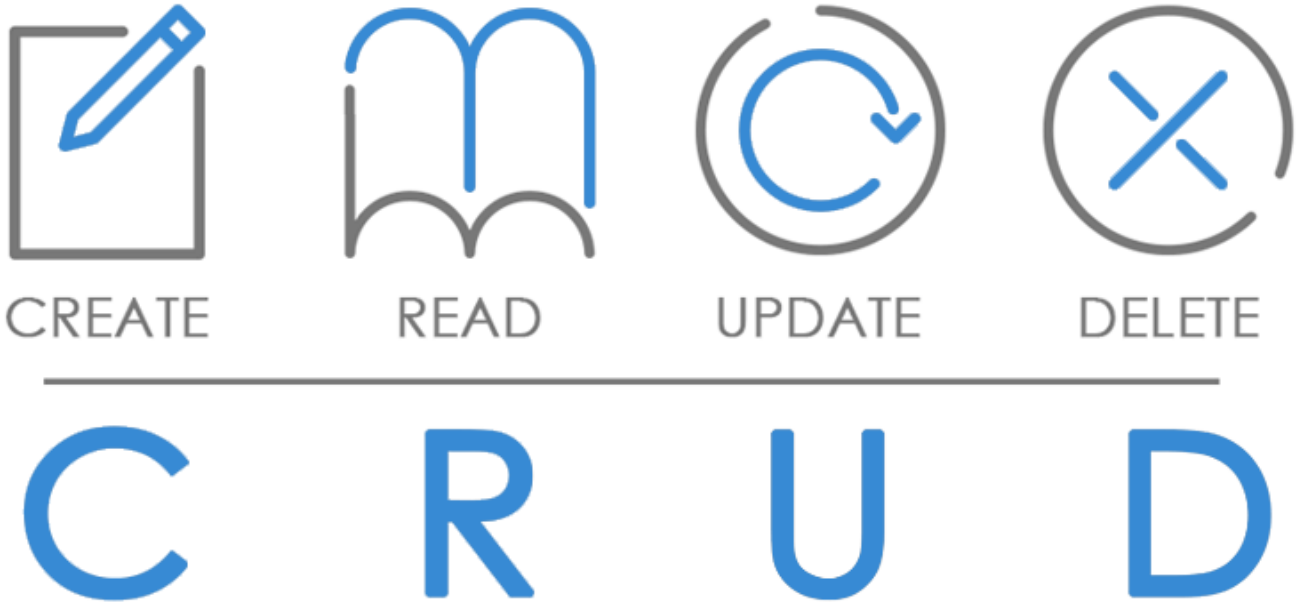
<Table_Name>: Tên của bảng cần xóa.

Ví dụ:

```
USE [QuanLySvBKAP]
DROP TABLE LOPHOC
```

Dữ liệu

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



Dữ liệu

Câu lệnh INSERT

Câu lệnh INSERT thêm một dòng mới vào bảng

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.

Dữ liệu

Câu lệnh INSERT

Ví dụ:

```
INSERT INTO MonHoc(ten_mon, ghi_chu) VALUES
('Java Core', N'Chương trình java cơ bản'),
('PHP', N'Dạy lập trình web với PHP '),
('Android', N'Lập trình di động Android'),
('PhoneGap', N'Lập trình ứng dụng di động đa nền tảng'),
('Windows Store', N'Lập trình ứng dụng Windows Store')
GO
```

Dữ liệu

Câu lệnh UPDATE

Câu lệnh UPDATE dùng để sửa dữ liệu trong bảng.

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.

Dữ liệu

Câu lệnh UPDATE

Ví dụ:

```
UPDATE LOPHOC SET
ten_lop = N'LTV Xuất sắc nhất Vịnh Bắc Bộ'
WHERE id_lop = 1
```

Dữ liệu

Câu lệnh DELETE

Câu lệnh DELETE xóa bỏ các dòng khỏi bảng

Cú pháp

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

Trong đó:

<Table_Name>: là tên của bảng mà các dòng sẽ được 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.

Dữ liệu

Câu lệnh DELETE

Ví dụ:

```
-- Xóa lớp học có ID = 2
DELETE LOPHOC WHERE id_lop = 2
```

Ràng buộc

Một ràng buộc (**constraint**) 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. SQL Server hỗ trợ các ràng buộc:

- **PRIMARY KEY**
- **UNIQUE**
- **FOREIGN KEY**
- **CHECK**
- **NOT NULL**

Ràng buộc

PRIMARY KEY (khóa chính):

- Mỗi bảng thường **có một khóa chính** gồm một cột hai 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 gía trị NULL**.

Ràng buộc

Ví dụ:

```
-- Tạo bảng thông thường
CREATE TABLE TblNhanVien(
id int IDENTITY(1,1) PRIMARY KEY,
manv varchar(8),
tennv nvarchar(128),
ngaysinh date,
gender bit
)
```

Ràng buộc

UNIQUE:

- Một ràng buộc UNIQUE được sử dụng để **đảm bảo các giá tri 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.
- Việc thêm **dữ liệu mà trùng lặp** sẽ gặp **lỗi** và **không thể thực hiện**.

Ràng buộc

Ví dụ:

```
-- Bảng có trường SoDienThoai và Email không được
trùng
CREATE TABLE NhanVienBKAP(
NhanVienID int PRIMARY KEY,
HoVaTen nvarchar(64) NOT NULL,
SoDienThoai varchar(11) UNIQUE,
ngaysinh date,
gender bit,
Email varchar(11) UNIQUE
)
```

Ràng buộc

FOREIGN KEY (khóa ngoại):

- 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**.

Ví dụ: có 2 bảng **LOAI_SAN_PHAM** và **SAN_PHAM**, sự ràng buộc giữa 2 bảng trên là *Sản Phẩm* sẽ thuộc về một *Loại Sản Phẩm* cụ thể. Xem mã nguồn sau:

Ràng buộc

```
-- Tạo bảng LOAI_SAN_PHAM
CREATE TABLE LOAI_SAN_PHAM(
ma_lsp int identity(1,1) PRIMARY KEY, -- Khóa chính, kiểu int, tự tăng
ten_loai nvarchar(50) NOT NULL, -- Không được để trống
trang_thai tinyint DEFAULT(1) -- Mặc định dữ liệu là 1
)

-- Tạo bảng SAN_PHAM
CREATE TABLE SAN_PHAM(
ma_sp int identity(1,1) PRIMARY KEY, -- Khóa chính
ten_sp nvarchar(200) NOT NULL, -- Không để trống
gia_nhap float NOT NULL DEFAULT(0), -- Không để trống, mặc định là 0
ma_lsp int FOREIGN KEY REFERENCES LOAI_SAN_PHAM(ma_lsp)
)
```

Ràng buộc khóa ngoại đảm bảo ma_lsp phải có bên bảng LOAI_SAN_PHAM

Ràng buộc

CHECK:

- Ràng buộc CHECK được sử dụng để **giới hạn các giá trị có thể được nhập** vào một cột.
- Ràng buộc CHECK **đảm bảo 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).

Ví dụ: bảng SAN_PHAM cần quy định ngày nhập phải đảm bảo lớn hơn hoặc bằng ngày giờ hiện tại (hệ thống). Xem mã nguồn sau:

Ràng buộc

```
-- Tạo bảng SAN_PHAM
CREATE TABLE SAN_PHAM(
ma_sp int identity(1,1) PRIMARY KEY, -- Khóa chính
ten_sp nvarchar(200) NOT NULL, -- Không để trống
mo_ta nvarchar(250),
gia_nhap float NOT NULL DEFAULT(0), -- Không để trống, mặc
định là 0
ngay_cap_nhat datetime CHECK(ngay_cap_nhat >= GETDATE()), --
Kiểm tra giá trị nhập luôn >= ngày hiện tại
ma_lsp int FOREIGN KEY REFERENCES LOAI_SAN_PHAM(ma_lsp)
)
```

Ràng buộc

Giá trị NULL / NOT NULL:

- 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 **trắng**, hoặc độ dài chuỗi bằng 0 (**như ' '**). Ví dụ: giá trị null trong cột Color của bảng Production.
- 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.

Ràng buộc

- 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.

	categoryid	categoryname	description
1	1	Beverages	Soft drinks, coffees, teas, beers, and ales
2	2	Condiments	NULL
3	3	Confections	Desserts, candies, and sweet breads
4	4	Dairy Products	NULL
5	5	Grains/Cereals	NULL
6	6	Meat/Poultry	Prepared meats
7	7	Produce	Dried fruit and bean curd
8	8	Seafood	Seaweed and fish

Ràng buộc

Ví dụ:

```
-- Tạo bảng LOAI_SAN_PHAM
CREATE TABLE LOAI_SAN_PHAM(
ma_lsp int identity(1,1) PRIMARY KEY, -- Khóa chính,
kiểu int, tự tăng
ten_loai nvarchar(50) NOT NULL, -- Không được để trống
trang_thai tinyint DEFAULT(1) -- Mặc định dữ liệu là 1
)
GO
```

Thuộc tính/Định nghĩa

DEFAULT:

- Một định nghĩa DEFAULT được sử dụng để gắn 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.

Thuộc tính/Định nghĩa

- 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**.

Thuộc tính/Định nghĩa

Ví dụ:

```
-- Tạo bảng SAN_PHAM
CREATE TABLE SAN_PHAM(
ma_sp int identity(1,1) PRIMARY KEY, -- Khóa chính
ten_sp nvarchar(200) NOT NULL, -- Không để trống
gia_nhap float NOT NULL DEFAULT(0), -- Không để trống,
mặc định là 0
luot_xem int NOT NULL DEFAULT(0), -- Không để trống, mặc
định là 0
)
GO
```

Thuộc tính/Định nghĩa

IDENTITY:

- 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**).

Thuộc tính/Định nghĩa

IDENTITY – các đặc điểm

- Cột có thuộc tính `IDENTITY` phải là các cột có kiểu dư 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/Định nghĩa

Ví dụ:

```
-- Tạo bảng SAN_PHAM
CREATE TABLE SAN_PHAM(
ma_sp int identity(1,1) PRIMARY KEY, -- Khóa chính
ten_sp nvarchar(200) NOT NULL, -- Không để trống
mo_ta nvarchar(250),
thong_tin ntext)
GO
```

Thuộc tính/Định nghĩa

UNIQUEIDENTIFIER:

- 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**

Thuộc tính/Định nghĩa

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

Thuộc tính/Định nghĩa

Ví dụ:

```
CREATE TABLE Khoa(
idkhoa int IDENTITY(1,1) PRIMARY KEY,
makhoa uniqueidentifier,
tenkhoa nvarchar(128)
)
INSERT INTO Khoa(makhoa, tenkhoa) VALUES
(NEWID(), 'Khoa hoc xa hoi')
SELECT * FROM Khoa
```

Tóm tắt bài học

- Hầu hết các bảng đều có một khóa chính (**primary key**), được tạo ra từ một hoặc nhiều cột của bảng đẻ 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.

Tóm tắt bài học

- Một constraint là một thuộc tính được gắn tới cột để áp dụng các nguyên tắc xử lý logic (**bussiness logic rule**) và đảm bảo tính toàn vẹn.
- Một ràng buộc **UNIQUE** được sử dụng để đảm bảo các giá tri 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.

Thank for watching!