Chương 2

Tạo và quản trị Cơ sở dữ liệu

Nội dung

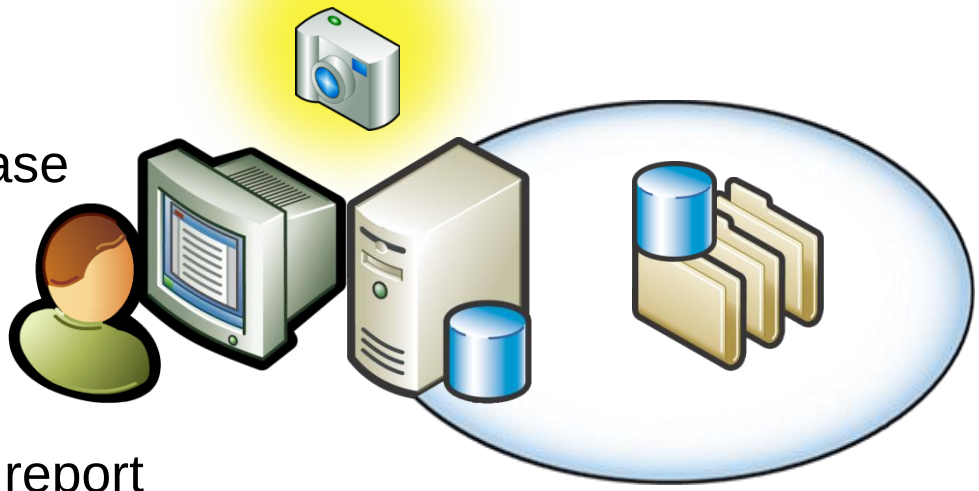
- Tạo và quản trị Database
- Các kiểu dữ liệu
- Tạo và quản lý bảng
- Ràng buộc toàn vẹn đơn giản
- Chỉ mục

Databases in SQL Server

- Database

  Storing data and other database objects

- Database Snapshot
  - Maintain historical data for report generation
  - Safeguard data against administrative error
  - Safeguard data against user error



SQL Server Enterprise Edition

Database in SQL Server

Overview of Database Objects

Tables	Indexes	Views	Programmability
• Data storage & Retrieval • Referential integrity	• Improves query performance • Clustered • Non-clustered	• Logical result sets • Based on SELECT queries	• Stored Procedures • Functions • Triggers • Constraints

Về mặt vật lý: một Database bao gồm hai hay nhiều hơn hai tập tin trên một hay nhiều đĩa. Chỉ thấy được bởi nhà quản trị và nó trong suốt đối với người sử dụng

Về mặt Logic: một database được xây dựng thành các thành phần mà được hiển thị với người dùng như Table, View, Procedure, …

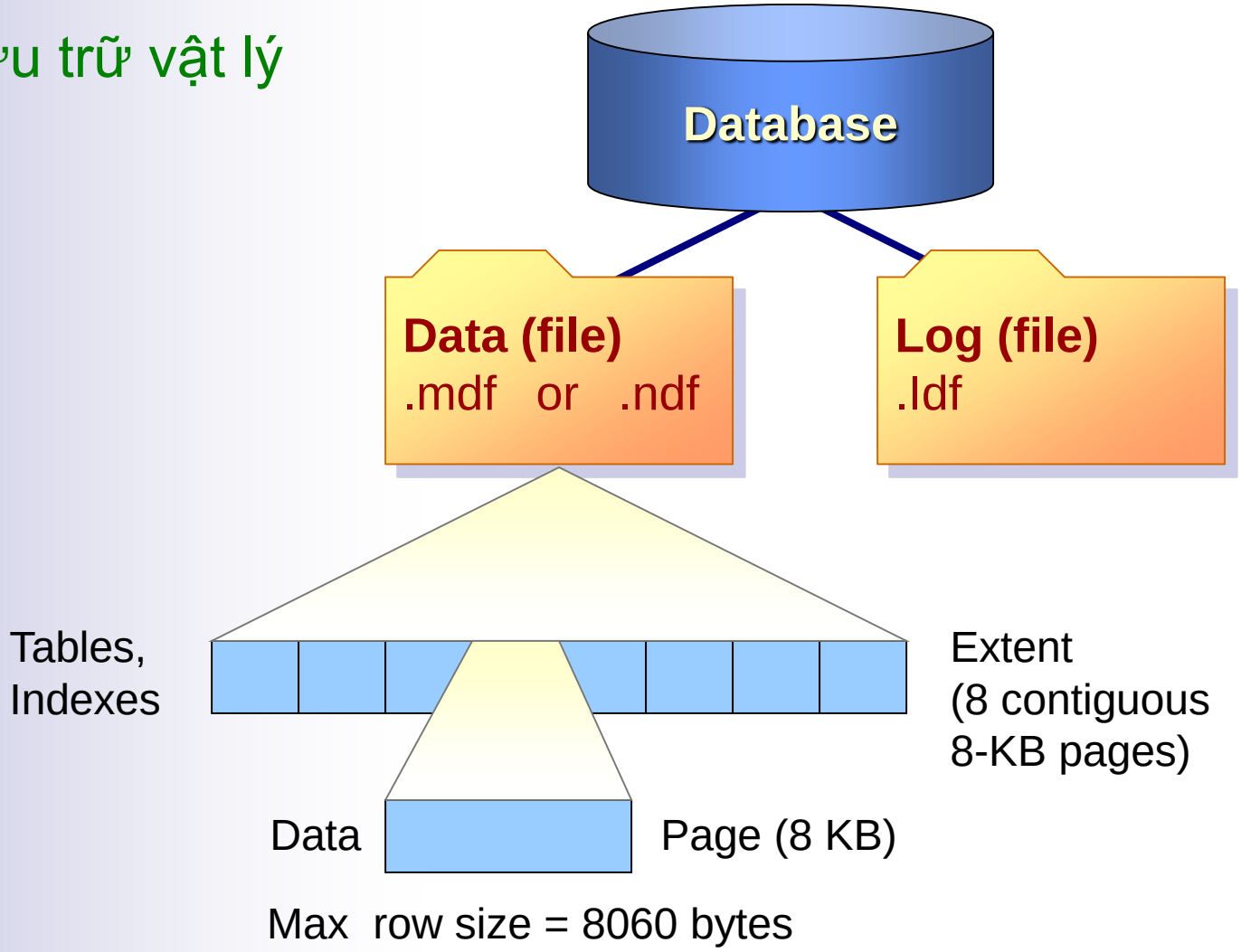
Database in SQL Server

Có tối thiểu 2 tập tin trong CSDL:

- **File dữ liệu cơ bản (Primary data file) (.mdf):** mỗi CSDL chỉ có duy nhất 1 file cơ bản (mặc định), dùng để ghi nhận lại tất cả những tập tin khác trong CSDL và lưu trữ dữ liệu.
- **Các file thứ cấp (Secondary data files) (.ndf) (tuỳ chọn):** một CSDL có thể có hay không có nhiều file thứ cấp, dùng để lưu các đối tượng của CSDL.
- **File nhật ký giao dịch (Transaction log file) (.ldf):** mỗi CSDL có từ 1 hay nhiều file nhật ký, dùng để chứa những thông cần thiết cho việc phục hồi tất cả những giao tác (transaction) trong CSDL.

Database in SQL Server

Lưu trữ vật lý

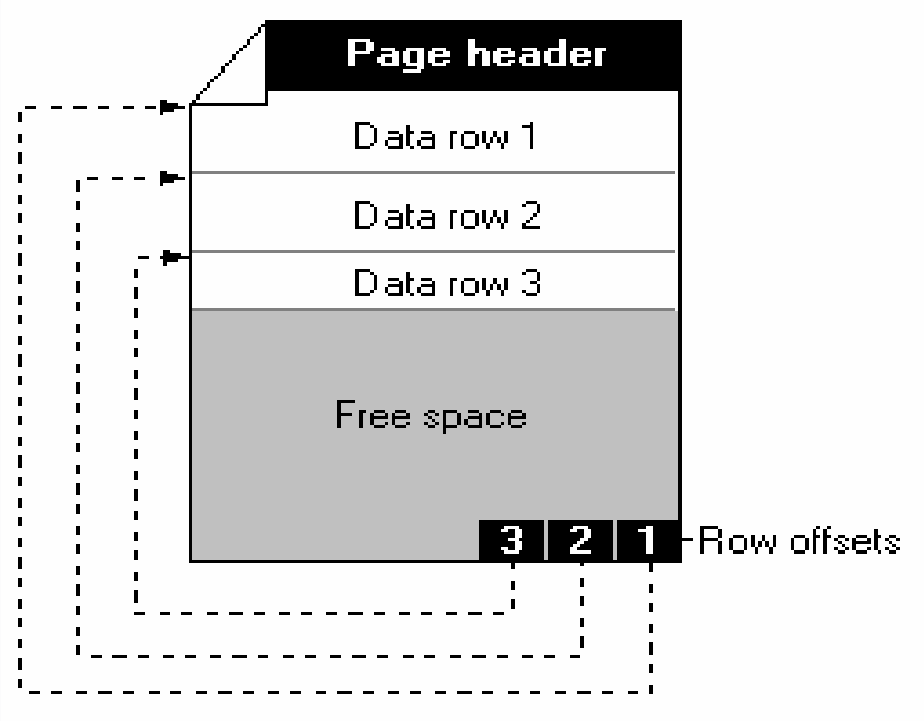


Database in SQL Server

- Đơn vị cơ bản để lưu trữ CSDL là trang (page)
- Trang là 1 khối 8KB nằm liên tục trên đĩa
- File log không chứa các trang, nó là 1 chuỗi các record
- Các trang dữ liệu chứa tất cả các dữ liệu của hàng (row), ngoại trừ kiểu dữ liệu text và image nằm trên các trang riêng.
- Các hàng (row) của bảng không thể quá lớn để kéo dài từ trang này sang trang khác, vì vậy mỗi hàng bị giới hạn không thể lớn hơn 8KB
- Các hàng dữ liệu nằm tuần tự trên mỗi trang ngay sau tiêu đề (header) của trang

Database in SQL Server

- Page header chiếm 96 byte chứa thông tin hệ thống như loại trang, số không gian còn trống ,…

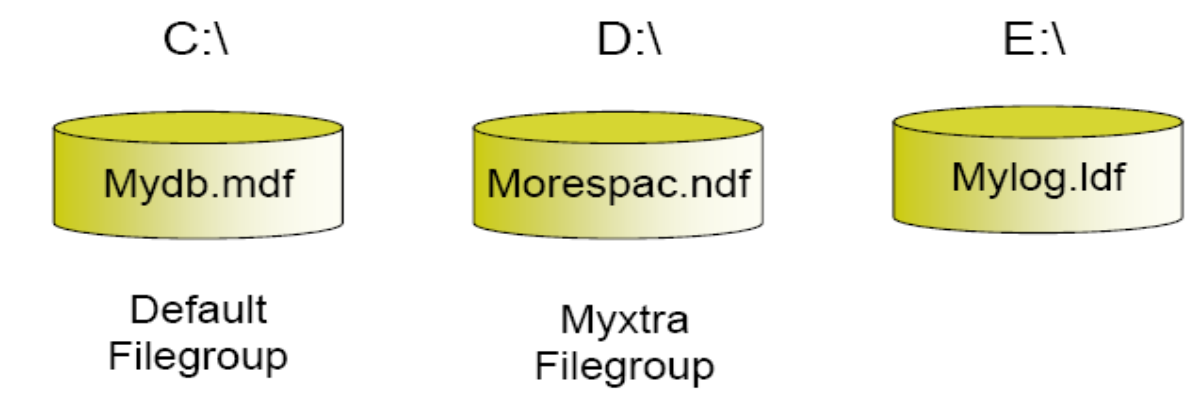


Database in SQL Server

- Các trang được tổ chức thành các extent
- Một extent là 1 tập hợp 8 trang liên tục nhau.
- Một CSDL có 16 extents (128 pages) trên 1 MB
- Extent đầu tiên của mỗi file CSDL được dùng riêng bởi SQL server để theo dõi việc phân phối dữ liệu trên đĩa

Files and Filegroups

- Khi tạo 1 CSDL, thì các file dữ liệu và log được tạo ra tại vị trí do ta xác định.
- Các file này có nằm trên những đĩa vật lý khác nhau để cải thiện việc thực thi của hệ thống.
- Filegroup có thể chứa 1 hay nhiều file. Một CSDL có thể được chứa trong 1 hay 1 số filegroup. Có 3 loại: Primary filegroup, user-define filegroups và default filegroup



Files and Filegroups

- Primary FileGroup: chứa file dữ liệu chính (.mdf) và bất cứ file thứ cấp nào (.ndf). Tất cả các bảng hệ thống phải nằm trong primary filegroup.
- User-defined filegroup: do người dùng xác định trong lệnh CREATE/ALTER DATABASE
- Default filegroup: là bất kỳ filegroup nào trong DB. Thường thì primary filegroup chính là default filegroup nhưng owner có quyền thay đổi. Tất cả bảng và index mặc định đều được tạo ra trong default filegroup.

Creating a New Database

- Cú pháp lệnh tạo CSDL :

```
CREATE DATABASE database_name
[ ON
[ < filespec > [ ,...n ] ]
[ , < filegroup > [ ,...n ] ]
]
[ LOG ON { < filespec > [ ,...n ] } ]
```

- Cú pháp Filespec:

```
(NAME = logical_name,
FILENAME = 'path\filename',
SIZE = size_in_MB,
MAXSIZE = size_in_MB | UNLIMITED,
FILEGROWTH = %_or_MB)
```

Creating a New Database

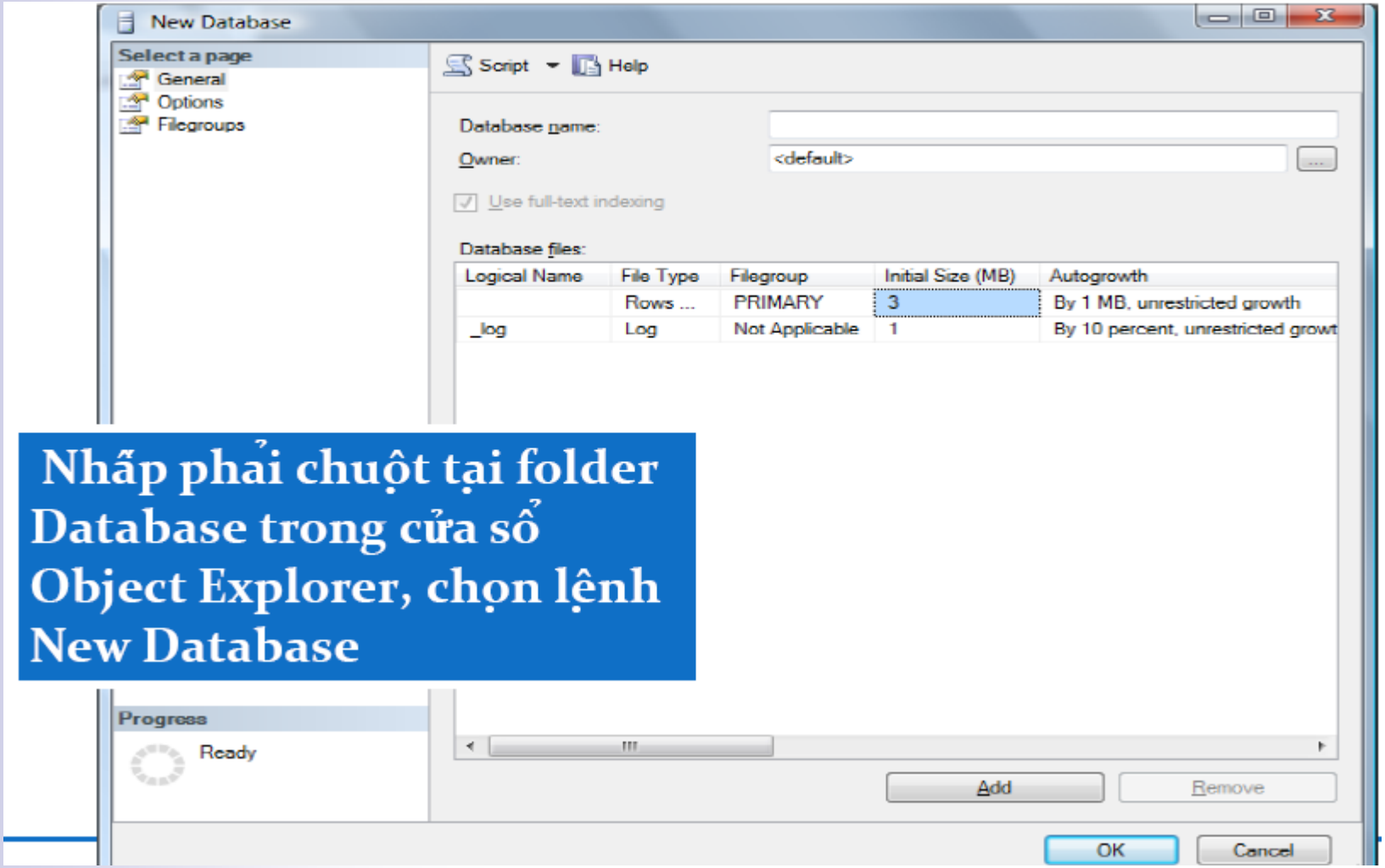
- Some arguments:
  - The name of the database
  - The size of the database
  - The files where the database will reside

```
CREATE DATABASE Sample
ON
  PRIMARY ( NAME=SampleData,
  FILENAME='c:\Program Files\..\..\Data\Sample.mdf',
  SIZE=10MB,
  MAXSIZE=15MB,
  FILEGROWTH=20%)
LOG ON
  ( NAME=SampleLog,
  FILENAME= 'c:\Program Files\..\..\Data\Sample.ldf',
  SIZE=3MB,
  MAXSIZE=5MB,
  FILEGROWTH=1MB)
COLLATE SQL_Latin1_General_Cp1_CI_AS
```

Creating a New Database

```
CREATE DATABASE Sales
ON PRIMARY
( NAME = Sales1_dat, FILENAME = 'D:\BTSQL\Sales1_dat.mdf',
    SIZE = 10, MAXSIZE = 50, FILEGROWTH = 15% ),
( NAME = Sales2_dat, FILENAME = 'D:\BTSQL\Sales2_dat.ndf',
    SIZE = 10, MAXSIZE = 50, FILEGROWTH = 15% ),
FILEGROUP SalesGroup1
( NAME = Sales3_dat, FILENAME = 'D:\BTSQL\Sales3_dat.ndf',
    SIZE = 10, MAXSIZE = 50, FILEGROWTH = 5 ),
( NAME = Sales4_dat, FILENAME = 'D:\BTSQL\Sales4_dat.ndf',
    SIZE = 10, MAXSIZE = 50, FILEGROWTH = 5 )
LOG ON
( NAME = 'Sales_log',
    FILENAME = 'D:\BTSQL\salelog.ldf', SIZE = 5MB,
    MAXSIZE = 25MB, FILEGROWTH = 5MB )
```

Creating a New Database (từ menu)



Attaching an Existing Database

```
CREATE DATABASE database_name
    ON <filespec> [ ,...n ]
    FOR { ATTACH [ WITH <service_broker_option> ]
        | ATTACH_REBUILD_LOG }
[;]
```

```
<service_broker_option> ::=
{
    ENABLE_BROKER
  | NEW_BROKER
  | ERROR_BROKER_CONVERSATIONS
}
```

Creating a Snapshot Database

```
CREATE DATABASE database_snapshot_name
    ON
        (
        NAME = logical_file_name,
        FILENAME = 'os_file_name'
        ) [ ,...n ]
    AS SNAPSHOT OF source_database_name
[;]
```

Managing Databases

- Managing Data and Log File Growth
- Monitoring and Expanding a Transaction Log
- Shrinking a Database or File
- Dropping a Database

Managing Database

Hiển thị thông tin DB

- Mở CSDL

  USE TenCSDL

  Ví dụ:

  use Sales

- Kiểm tra sự tồn tại của CSDL

  sp_helpdb TenCSDL

- Kiểm tra không gian sử dụng của CSDL

  sp_spaceused

Managing Databases

Cú pháp lệnh thay đổi cấu trúc CSDL

```
ALTER DATABASE database_name
    ADD FILE filespec [TO FILEGROUP filegroup_name]
    ADD LOG FILE filespec
    | REMOVE FILE logical_filename
    | ADD FILEGROUP filegroup_name
    | REMOVE FILEGROUP filegroup_name
    | MODIFY FILE filespec
    | MODIFY FILEGROUP filegroup_name
    filegroup_property
    |SET optionspec [WITH termination]
```

Managing Databases

Ví dụ thay đổi cấu trúc CSDL

a) Chỉnh sửa Size của tập tin

```
ALTER DATABASE Sales
MODIFY FILE (NAME = 'Sales_log', size =10MB)
```

b) Bổ sung thêm một tập tin dữ liệu

```
ALTER DATABASE Sales
ADD File (Name =Sales5_dat,
    Filename ='D:\BTSQL\Sales5_dat.mdf',SIZE =10 MB,
    Maxsize =20MB)
```

Managing Data and Log File Growth

- Using Automatic File Growth
- Expanding Database Files
- Adding Secondary Database Files

```
ALTER DATABASE Sample
   MODIFY FILE ( NAME = 'SampleLog',
   SIZE = 15MB)
GO

ALTER DATABASE Sample
ADD FILE
  (NAME = SampleData2,
   FILENAME='c:\Program Files\..\..\
      Data\Sample2.ndf',
   SIZE=15MB,
   MAXSIZE=20MB)
GO
```

Managing Database

- Xem các thuộc tính của CSDL

```
SELECT
    DATABASEPROPERTYEX('databasename',
    'property')
```

- Property: IsAutoShrink, IsCloseCursorsOnCommitEnabled, Recovery, Updateability, UserAccess
- Trị trả về: 1 = TRUE, 0 = FALSE, NULL = Input not valid
- select DATABASEPROPERTYEX('Sales', 'IsAutoShrink')

Managing Data and Log File Growth

Thay đổi thuộc tính DB

ALTER DATABASE database_name
SET option [, status]

Option

AUTO_SHRINK
CURSOR_CLOSE_ON_COMMIT
RECOVERY FULL | BULK_LOGGED | SIMPLE
SINGLE_USER | RESTRICTED_USER | ULTI_USER
READ_ONLY | READ_WRITE

Example:

ALTER DATABASE Sales
SET Read_Only

Managing Data and Log File Growth

- Đổi tên cơ sở dữ liệu:

  sp_renamedb [@dbname =] 'old_name', [@newname =] 'new_name‘

VD: Sp_ReNamedb 'Sales', 'Banhang'

Managing Data and Log File Growth

Xóa cơ sở dữ liệu:

- Khi 1 CSDL bị xóa thì tất cả các file vật lý của nó sẽ bị xóa
- **Cú pháp:**

  DROP DATABASE *database_name*

- **Ví dụ:**

  Drop Database Banhang

Chú ý: Không thể xóa các CSDL master, model, tempdb

Monitoring and Expanding a Transaction Log

- Monitoring the log
- Monitoring situations that produce extensive log activity
  - Mass loading of data into indexed table
  - Large transactions
  - Performing logged text or image operations
- Expanding the log when necessary

Outline

- √ Data Definition Language
- √ Managing Databases
- Data Types
- Managing Tables

System Data Types

Có 2 nhóm:

- **System-Supplied datatype:** Các kiểu dữ liệu cơ bản được hỗ trợ bởi SQL Server.
- **User-defined datatype:** Các kiểu dữ liệu của người dùng tự định nghĩa dựa trên các kiểu dữ liệu cơ bản.

System Data Types

Các kiểu dữ liệu cơ bản:

Loại	Kiểu dữ liệu cơ sở	Kích cỡ	Vùng giá trị	Mô tả
Binary	Binary	8 KB	"0"…"9", "a"..."f", "A"..."F"	Chứa các bit thông tin
	Varbinary	8 KB	"0"…"9", "a"..."f", "A"..."F"	
	Image	2^31 -1 bytes	2^31 –1 bytes	Dữ liệu hình ảnh
Character	Char	255 bytes	1..8000 ký tự	Ký tự hoặc chuỗi
	Varchar	255 bytes	1..8000 ký tự	Ký tự hoặc chuỗi
	Text	2147483647 bytes	2^31-1 ký tự (2147483647)	Ký tự hoặc chuỗi
Date and Time	Datetime	8 bytes	01/01/1753->31/12/9999	Chuỗi biểu diễn ngày giờ
	Smalldatetime	4 bytes	1/1/1900 -> 6/6/2079	Chuỗi biểu diễn ngày giờ
Decimal	Decimal	17 bytes	-10^38-1 -> 10^38-1	Số thực
	Numeric	17 bytes	-10^38-1 -> 10^38-1	Số thực
Foating point	Float	8 bytes	-1.79E+308 -> 1.79E+308	Số thực
	Real	4 bytes	-3.40E+38 ->3.40E+38	Số thực
Integer	Bigint	8 bytes	-2^63 -> 2^63	Số nguyên

System Data Types

	Int	4 bytes	-2^31 -> 2^31-1	Số nguyên
	Smallint	2 bytes	-2^15 -> 2^15-1	Số nguyên
	Tinyint	1 bytes	0..255	Số nguyên
Monetary	Money	8 bytes	-2^63 -> 2^63-1	Dữ liệu tiền tệ
	Smalmoney	4 bytes	-214748.3648 -> 214748.3648	Dữ liệu tiền tệ
Special	Bit	1 bytes	0 hoặc 1	Dữ liệu có một trong hai trạng thái 0 hoặc 1
	Cursor	Kiểu DL cho biến hoặc giá trị trả về của procedure, tham chiếu đến 1 mẫu tin		
	Timestamp	8 bytes	Chuỗi có dạng: 0x000000100000a90	Theo dõi mẫu tin nào bị thay đổi dữ liệu
	Uniqueidentifier	16 bytes	Số thập lục phân	
	SQL_variant	Là kiểu dữ liệu có thể chứa bất kỳ loại dữ tùy ý của SQL Server ngoại trừ **text**, **ntext**, **image**, and the **timestamp** data type		
	Table			
Unicode	Nchar		4000 ký tự	Ký tự hoặc chuỗi
	Nvarchar		4000 ký tự	Ký tự hoặc chuỗi
	Ntext		2^30-1 ký tự	Ký tự hoặc chuỗi

User-defined Data Type

Tạo một User-Defined Data Type

- Dùng thủ tục hệ thống ***sp_addtype*** để tạo một user-defined data type.

sp_addtype *type*, *system_data_type* [,'NULL' | 'NOT NULL']

- Ví dụ 1: Tạo kiểu dữ liệu tên là **isbn** với kiểu dữ liệu cơ bản là **smallint** và **không chấp nhận giá trị Null**

```
EXEC sp_addtype isbn, 'smallint', 'NOT NULL'
```

User-defined Data Type

- **Ví dụ 2:** Tạo kiểu dữ liệu tên là **zipcode** với kiểu dữ liệu cơ bản là **char**, **độ dài tối đa là 10 và chấp nhận giá trị Null**

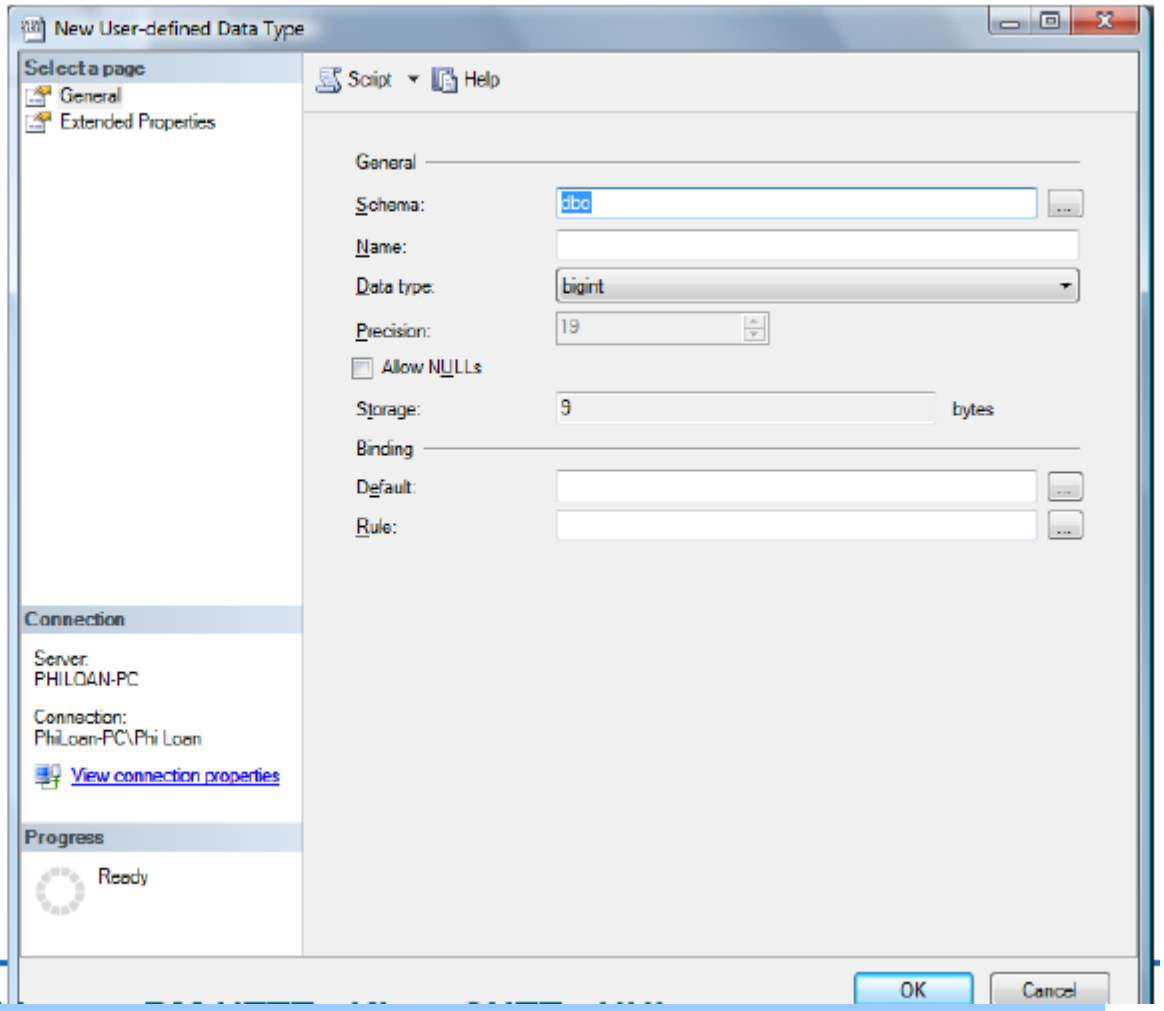
```
EXEC sp_addtype zipcode, 'char(10)', NULL
```

- Ví dụ 3: Tạo kiểu dữ liệu tên là **longstring** với kiểu dữ liệu cơ bản là **varchar**, **độ dài tối đa là 63** và **chấp nhận giá trị Null**

```
EXEC sp_addtype longstring, 'varchar(63)', NULL
```

User-defined Data Type – Tạo từ menu

Nhắp phải tại Programmability \Types\User-Defined Data Types, chọn New User Defined Data Type…)



User-defined Data Type

Xem các user-defined data types trong CSDL hiện hành:

- Dùng thủ tục **sp_help** hoặc truy vấn trong **information_schema.domains**
- Ví dụ: Use Sales

  Sp_help

  hoặc

```
SELECT domain_name, data_type, character_maximum_length
FROM information_schema.domains
ORDER BY domain_name
```

User-defined Data Type

- **Xoá một User-Defined Data Type**: dùng thủ tục hệ thống *sp_droptype để xóa một* user-defined data type từ bảng systypes. Một user-defined data type không thể xóa được nếu nó được tham chiếu bởi các bảng và những đối tượng khác.
- **Cú pháp:** **sp_droptype *type***
- **Ví dụ:**

```
EXEC sp_droptype isbn
```

Bảng dữ liệu - Table

- Bảng là một đối tượng của CSDL được dùng để lưu trữ dữ liệu.
- Dữ liệu trong bảng được tổ chức thành các hàng (rows) và cột (columns).
- Mỗi hàng trong bảng biểu diễn một bản ghi (record) duy nhất. Mỗi cột biểu diễn một thuộc tính (attribute).
- **Tên cột trong 1 bảng không được trùng nhau** nhưng cho phép **tên cột có thể trùng nhau trong những bảng khác nhau** của cùng 1 CSDL.
- SQL Server cho phép:
  - Tối đa 2 triệu bảng trong 1 CSDL.
  - Tối đa 1024 cột trong 1 bảng
  - Tối đa 8060 bytes trong 1 hàng

Bảng dữ liệu - Table

- Các bước tạo một bảng
  - **Bước 1:** Xác định kiểu dữ liệu của các cột.
  - **Bước 2:** Xác định các cột có thể hoặc không thể có giá trị rỗng (*null value*).
  - **Bước 3:** Xác định các cột phải có các giá trị duy nhất.
  - **Bước 4:** Xác định khóa chính – khóa ngoại.
  - **Bước 5:** Xác định các giá trị mặc định.
  - **Bước 6:** Xác định các ràng buộc trên các cột (mô tả miền trị).
  - **Bước 7:** Tạo bảng và các chỉ mục của bảng.

Tạo bảng - CREATE TABLE

```
CREATE TABLE
   [ database_name.[ owner ] .|owner.] table_name
   ({ < column_definition >
         |column_name AS
   computed_column_expression
         | < table_constraint > ::= [ CONSTRAINT
         constraint_name ] }
         [ { PRIMARY KEY | UNIQUE } [ ,...n ]
   )
[ ON { filegroup | DEFAULT } ]
[ TEXTIMAGE_ON { filegroup | DEFAULT } ]
```

Tạo bảng - CREATE TABLE

Cú pháp

```
CREATE TABLE <Table_Name>
(<Column_Name>  <Data_Type>,….)
```

Ví dụ

```
CREATE TABLE Sanpham
( MaSP CHAR(5), TenSP VARCHAR(15),  Dvt
VARCHAR(10), DonGia SMALLMONEY, SLTon INT
)
```

Tạo bảng - CREATE TABLE

IDENTITY [(*seed* , *increment*)]

- Tạo giá trị gia tăng duy nhất cho 1 cột, và cột này thường được dùng khoá chính cho bảng.
- Giá trị được gán thường là các kiểu dữ liệu sau: **tinyint**, **smallint**, **int**, **bigint**, **decimal(p,0)**, hay **numeric(p,0).**
- Trong mỗi bảng chỉ cho phép 1 cột là identity mà thôi.
- *Seed: là giá trị đầu tiên được tạo*.
- *Increment:là bước tăng để tạo ra giá trị kế tiếp.*
- Giá trị mặc định thường là (1,1).

Tạo bảng - CREATE TABLE

Cú pháp : Tạo cột có giá trị phát sinh tự động

```
CREATE TABLE <Table_Name>
(<Column_Name>  <Data_Type>
INDENTITY(seed[, Increment]) NOT NULL….)
```

Ví dụ

```
CREATE TABLE NhaCungCap
(MaNCC int Identity NOT NULL Primary key,
TenNCC VarChar(25))
```

Tạo bảng - CREATE TABLE

Cột tính toán - Computed column

- ***Cú pháp:***

  column_name AS computed_column_expression

- Là một cột ảo không được lưu trữ vật lý trong bảng. Nó được tính toán dựa vào các cột khác trong cùng bảng thông qua 1 biểu thức. Ví dụ : cost AS price * qty.
- Được dùng trong mệnh đề SELECT, WHERE, hay ORDER BY. Không thể dùng trong lệnh INSERT hay UPDATE
- Được dùng như giá trị khóa trong chỉ mục hay 1 phần của các ràng buộc PRIMARY KEY hay UNIQUE nếu giá trị của nó được định nghĩa bởi 1 biểu thức xác định và kiểu dữ liệu của giá trị trả về hợp lệ.
- Ví dụ: Cột tính toán a+b có thể được dùng làm chỉ mục nhưng a+DATEPART(dd, GETDATE()) không thể dùng làm chỉ mục

Tạo bảng - CREATE TABLE

Cột tính toán - Computed column

- **Ví dụ 1**

```
CREATE TABLE CtHoadon
(    SoHd int NOT NULL,
     MaHang char(5) NOT NULL,
     SoLuong int NOT NULL,
     DonGia money,
     ThanhTien AS SoLuong*DonGia
)
```

Tạo bảng có khóa chính -Primary Key

CREATE TABLE TableName

(columname datatype [,…],[CONSTRAINT *constraint_name*]

PRIMARY KEY [CLUSTERED|NONCLUSTERED]

{(column [ASC |DESC][,…,n])}

[WITH FILLFACTOR = fillfactor]

[ON {filegroup|DEFAULT}]

Ràng buộc Primary Key

- **Ví dụ 1**

```
CREATE TABLE DEAN
(
        MADA smallint PRIMARY KEY
        CLUSTERED NOT NULL,
        TENDA varchar(50) NOT NULL
        DEFAULT 'Chua '
)
```

Ràng buộc Primary Key

Ví dụ 4:

```
CREATE TABLE PHANCONG (
    MaNv int NOT NULL, MaDA smallint, SoNc int,
    primary key (MaNv,MaDa))
```

Ràng buộc Foreign key

Định nghĩa FOREIGN KEY CONSTRAIT khi tạo bảng

CREATE TABLE TableName

(columnName datatype [,…],

[CONSTRAINT constraintName]

FOREIGN KEY[(column[,..n])]

REFERENCES *ref_table* [**(** **r***ef_column [,..n])]***)**]

[ON DELETE { CASCADE | NO ACTION }]

[ON UPDATE { CASCADE | NO ACTION }]

[NOT FOR REPLICATION]

Ràng buộc Foreign key

ON UPDATE|DELETE {CASCADE | NO ACTION}

- Xác định hành động cần phải thực hiện cho 1 hàng trong bảng đang tạo nếu hàng đó có quan hệ tham chiếu và hàng tham chiếu bị xoá khỏi bảng chính. Mặc định là NO ACTION.
- CASCADE: dùng để xác định là hàng sẽ bị cập nhật/xoá khỏi bảng tham chiếu nếu hàng đó bị cập nhật/xóa khỏi bảng chính
- NO ACTION: SQL Server sẽ đưa ra thông báo lỗi và việc xoá hàng trên bảng chính sẽ bị từ chối.

Ràng buộc Foreign key

- **Ví dụ 1**

```
CREATE TABLE VITRI
(MaVt int Primary key, DiaChi varchar(40))

CREATE TABLE PhongBan
   (        MaPb int primary key,
            TenPb varchar(30),
            MaVt int REFERENCES VITRI(MaVt)
)
```

Ràng buộc Foreign key

- **Ví dụ 2**

```
CREATE TABLE NHANVIEN (
        MaNv  CHAR(9) NOT NULL,
        HoNv VARCHAR(15) NOT NULL,
        TenNv VARCHAR(15) NOT NULL,
        NgSinh DATETIME, DiaChi VARCHAR(30),
        Phai CHAR(1), Ma_Nql CHAR(9),
        Phg INT  NOT NULL,
        CONSTRAINT Nv_PK PRIMARY KEY (MaNv),
        CONSTRAINT Nv_Fk FOREIGN KEY (Phg)
        REFERENCES PHONGBAN(MaPb))
```

Ràng buộc Check

```
CREATE TABLE Orders (
OrderID int IDENTITY (1, 1) NOT NULL,
CustomerID nchar (5) CHECK (CustomerID LIKE '[A-
    Z][A-Z][A-Z][A-Z][A-Z]'),
EmployeeID int NULL, OrderDate datetime NULL
CHECK (OrderDate BETWEEN '01/01/70' AND
    GETDATE()),
RequiredDate datetime NULL, ShipVia int NULL
CHECK (ShipVia IN (1, 2, 3, 4)),
Freight money NULL CHECK (Freight>=0),
ShipCountry nvarchar (15),
CHECK (RequiredDate>OrderDate))
```

Ràng buộc Check

Ví dụ 4:

```
CREATE TABLE PHANCONG(
        ma_nvien CHAR(9) NOT NULL,
        soda smallint NOT NULL,
        thoigian DECIMAL(3,1) NOT NULL,
        PRIMARY KEY (ma_nvien, soda),
        FOREIGN KEY (ma_nvien) REFERENCES
          NHANVIEN,
        FOREIGN KEY (soda) REFERENCES
          DEAN(mada),
        CHECK (thoigian >= 0))
```

Ràng buộc Default

- Cú pháp: DEFAULT *constant_expression*
- Default dùng để xác định giá trị "có sẵn" được gán cho 1 cột khi thêm 1 bản ghi mới vào bảng.
- DEFAULT có thể áp dụng cho bất kỳ cột nào trong bảng ngoại trừ cột có kiểu timestamp hay có thuộc tính IDENTITY.
- *constant_expression: c*hỉ có giá trị hằng như chuỗi ký tự, hàm hệ thống, hay giá trị NULL.

```
CREATE TABLE <TableName>
(<Column_Name>  <DataType>
DEFAULT (<expresion>))
```

Ràng buộc Default

Ví dụ 1

```
CREATE TABLE HoaDon
(MaHD int, LoaiHD Char(1) DEFAULT 'X',
NgayLap DateTime NOT NULL)
```

Tạo bảng - CREATE TABLE

Khai báo Filegroup chứa Table

```
CREATE TABLE <Table_Name>
(<Column_Name>  <Data Type>,…)
ON FileGroupName
```

Ví dụ

```
CREATE TABLE KH
(MaKh int Identity(1000,1) NOT NULL, TenKH Varchar(40))
ON SalesGroup1
```

Sửa cấu trúc bảng

Cú pháp

```
ALTER TABLE <table_name>
{ALTER COLUMN <column_name>
<new_data_type>}
| {ADD [<column_name> <data_type>]}
| {DROP COLUMN <column_name>}
```

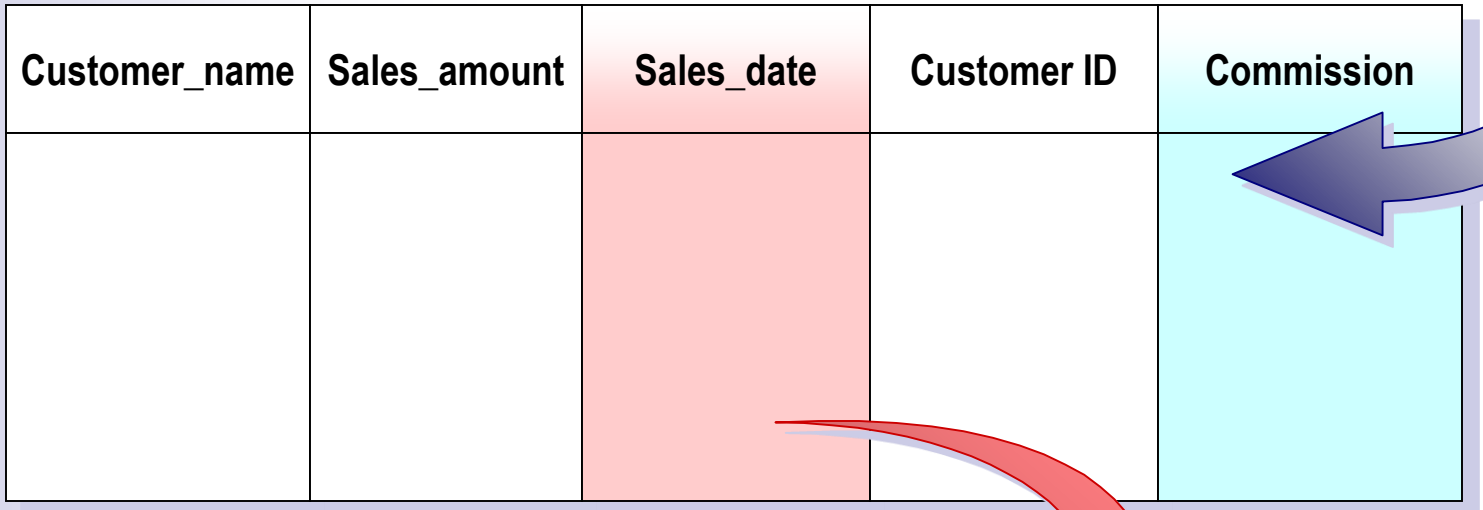
Ví dụ: thêm cột

```
ALTER TABLE SanPham
ADD NgayNhap SmallDateTime
```

Example

ADD

```
ALTER TABLE CategoriesNew
    ADD Commission money null
```



DROP

```
ALTER TABLE CategoriesNew
    DROP COLUMN Sales_date
```

Sửa cấu trúc bảng

Ví dụ thêm cột

Cú pháp

```
ALTER TABLE <table_name>
{ALTER COLUMN <column_name> <new_data_type>}
| {ADD [<column_name> <data_type>]}
| {DROP COLUMN <column_name>}
```

Ví dụ: sửa kiểu dữ liệu cho cột

```
ALTER TABLE SanPham
ALTER COLUMN NgayNhap DateTime NOT NULL
```

Sửa cấu trúc bảng

Syntax

```
ALTER TABLE <table_name>
{ALTER COLUMN <column_name> <new_data_type>}
| {ADD [<column_name> <data_type>]}
| {DROP COLUMN <column_name>}
```

Example

```
ALTER TABLE Sanpham
DROP COLUMN NgayNhap
```

Xóa bảng khỏi CSDL

Cú pháp

```
DROP TABLE <Table_Name>
```

Ví dụ

```
DROP TABLE SanPham
```

Bảng tạm

- Bảng tạm được chứa trong CSDL TempDb và được xóa một cách tự động khi không còn sử dụng nữa.
- Có hai loại:
  - Bảng tạm cục bộ
  - Bảng tạm toàn cục

Bảng tạm

Bảng tạm cục bộ:

- Có một dấu # là ký tự đầu tiên trong tên bảng tạm.
- Chỉ hiện thị đối với nối kết hiện hành dành cho người sử dụng.
- Được xóa khi người dùng ngắt nối kết với các thể hiện của SQL Server.

Ví dụ: Tạo bảng tạm là #MyLocalTemTable

```
CREATE TABLE #MyLocalTemTable
(       ID INT Primary key,
        ColA Varchar(30) NULL)
```

Bảng tạm

Bảng tạm toàn cục:

- Có hai dấu ## là 2 ký tự đầu tiên trong tên bảng tạm.
- Chỉ hiển thị đối với bất kỳ người sử dụng nào sau khi chúng được tạo.
- Được xóa khi tất cả người dùng đang tham chiếu table ngắt kết nối với SQL Server.

Ví dụ: Tạo bảng tạm là ##MyLocalTemTable

```
CREATE TABLE ##MyLocalTemTable
(       ID INT Primary key,
        ColA Varchar(30) NULL)
```

Cập nhập nội dung Table

Cú pháp: Thêm dòng

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

Insert into cthoadon values(2,'b', 10,100)

Cú pháp: Thay đổi dữ liệu các dòng

```
UPDATE <table_name>
SET <column_name = value>
WHERE <condition>
```

Update cthoadon set dongia = dongia+10/100*dongia

Cú pháp: Xóa dòng

```
DELETE FROM <table_name> WHERE <condition>
```

Delete from cthoadon where sohd =2

Xem Tables

Cú pháp: Xem thông tin Table

```
sp_help <table_name>
```

sp_help cthoadon

Cú pháp: Xem dữ liệu Table

```
SELECT <select_list> FROM <table_name>
```

Select * From cthoadon

Toàn vẹn dữ liệu

- TVDL là đề cập đến trạng thái của tất cả các giá trị dữ liệu lưu trữ trong CSDL là đúng. Nếu dữ liệu không đúng mà đã được lưu trữ trong CSDL thì gọi là vi phạm TVDL.
- Các loại ràng buộc toàn vẹm: Not Null, Default, Identity, Constraints, Rule, Triggers, Indexs.
- Định nghĩa ràng buộc:

  Create Table…: Định nghĩa trong lúc thiết kế.

  Alter Table…: Định nghĩa trong khi hiệu chỉnh bảng.
- Kiểm tra /xem các toàn vẹn dữ liệu:

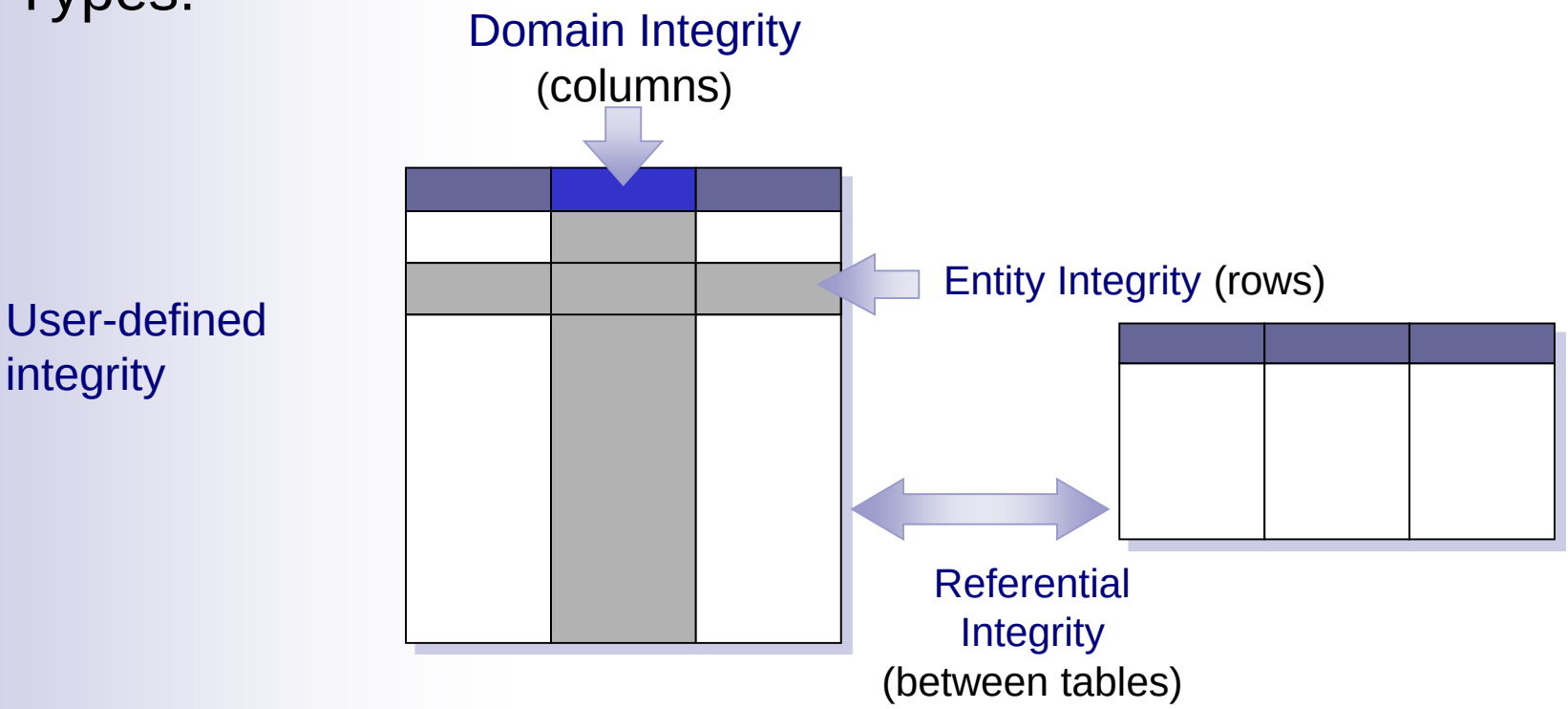
  Sp_HelpConstraint <Tên Table>
- Xóa toàn vẹn dữ liệu:

  ALTER TABLE <TenTable>

  DROP CONSTRAINT <Ten Constrant>

Toàn vẹn dữ liệu

- Defining the quality of the data in the database.
- Types:



Định nghĩa NULL / NOT NULL

- Giá trị NULL dùng để chỉ các giá trị chưa biết, hay sẽ được bổ sung sau. Nó khác với giá trị rỗng (empty) hay zero. Hai giá trị null không được xem là bằng nhau. Khi so sánh hai giá trị null, hay 1 giá trị null với 1 giá trị khác thì kết quả trả về sẽ là unknown.
- Ví dụ: số phone của khách hàng hiện tại chưa có, nhưng có thể sẽ được bổ sung này. Số phone sẽ có giá trị là null.
- Để kiểm tra giá trị null trong lệnh truy vấn, phải dùng toán tử IS NULL hay IS NOT NULL trong mệnh đề WHERE.
- Mặc định các cột hay kiểu dữ liệu của người dùng thường không có giá trị NULL.

Định nghĩa NULL / NOT NULL

- **Ví dụ:**

```
CREATE TABLE San_Pham
(       Masp            smallint NOT NULL,
        Tensp           char(20) NOT NULL,
        Mota            char(30) NULL,
        Gia             smallmoney NOT NULL
)
```

Ràng buộc Default

- Cú pháp: DEFAULT *constant_expression*
- Default dùng để xác định giá trị "có sẵn" được gán cho 1 cột khi thêm 1 bản ghi mới vào bảng.
- DEFAULT có thể áp dụng cho bất kỳ cột nào trong bảng ngoại trừ cột có kiểu timestamp hay có thuộc tính IDENTITY.
- *constant_expression: c*hỉ có giá trị hằng như chuỗi ký tự, hàm hệ thống, hay giá trị NULL.

```
CREATE TABLE <TableName>
(<Column_Name>  <DataType>
DEFAULT (<expresion>))
```

```
ALTER TABLE tablename
ADD [ CONSTRAINT constraintname ]
DEFAULT expression FOR columnname
```

Ràng buộc Default

Ví dụ 1

```
CREATE TABLE HoaDon
(MaHD int, LoaiHD Char(1) DEFAULT 'X', NgayLap
DateTime NOT NULL)
```

```
ALTER TABLE HoaDon
ADD DEFAULT Getdate() FOR NgayLap
```

Hay

```
ALTER TABLE HoaDon
ADD CONSTRAINT Ngay_DF DEFAULT
Getdate() FOR NgayLap
```

Sử dụng Default

- Sau khi tạo được DEFAULT, nó cần được gắn kết vào 1 cột hay kiểu dữ liệu người dùng.

  sp_bindefault default_name, object_name [, FUTUREONLY]

- Xóa gắn kết default làm cho nó không còn áp dụng được vào cột của bảng hay kiểu dữ liệu người dùng.

  sp_unbindefault object_name [, FUTUREONLY]

Sử dụng Default

- **Ví dụ default và ràng buộc vào cột của bảng**

```
CREATE TABLE bang (c1 int)
GO
CREATE DEFAULT phonedflt AS 'unknown'
GO
EXEC sp_bindefault 'phonedflt', 'bang.c1'
GO
EXEC sp_unbindefault 'bang.c1'
GO
DROP DEFAULT phonedflt
```

Sử dụng Default

- Ví dụ default và ràng buộc vào kiểu dữ liệu người dùng

```
sp_addType typCity, 'char(15)'
go
CREATE DEFAULT defCity AS 'Oakland'
go
sp_bindefault defCity, 'typCity'
```

- Ví dụ

```
CREATE TABLE jobs
(       job_id smallint IDENTITY(1,1) ,
        job_desc varchar(50) NOT NULL
        DEFAULT 'New Position - title not formalized yet'
)
```

Ràng buộc Default

Xoá Default - mặc định

DROP DEFAULT { *default* } [**,**...*n*]

Hay

ALTER TABLE <TenTable>
DROP CONSTRAINT <TenDefault>

- Lệnh DROP có thể xóa cùng lúc nhiều default
- Ví dụ:

DROP DEFAULT phonedflt

Hay

ALTER TABLE Hoadon
DROP CONSTRAINT Ngay_DF

Ràng buộc Check

```
CREATE TABLE <Table_Name>
(<Column_Name>  <Data_Type>[,…] CONSTRAINT
ConstraintName] CHECK (NOT FOR REPLICATION]
<Logical expresion>),….)
```

- *Qui định nhập dữ liệu phải thỏa mãn điều kiện của biểu thức* <Logical expresion>.
- <Logical expresion>*:* biểu thức với các toán tử số học, toán tử quan hệ hay từ khoá IN, LIKE, BETWEEN.

```
ALTER TABLE <Table_Name>
[WITH CHECK | WITH NOCHECK] ADD
[CONSTRAINT ConstraintName]
CHECK (NOT FOR REPLICATION] <Logical expresion>),….)
```

Ràng buộc Check

- **Ví dụ 1:**

  CREATE TABLE NhanVien

  (MaNV char(4) CHECK (Manv LIKE '[0-9][0-9][0- 9][0-9]'), Hoten Varchar(40), LCB int CHECK (LCB BETWEEN 0 AND 50000), HSPC real, Thanhpho varchar(10) CONSTRAINT chkCity CHECK(Thanhpho IN ('Berkeley', 'Boston', 'Chicago', 'Dallas')))

- **Ví dụ 2:**

  ALTER TABLE Nhanvien

  ADD CONSTRAINT NV_HSPC

  CHECK (HSPC>=0.1 AND HSPC<0.5)

Ràng buộc Check

```
CREATE TABLE Orders (
OrderID int IDENTITY (1, 1) NOT NULL,
CustomerID nchar (5) CHECK (CustomerID LIKE '[A-
    Z][A-Z][A-Z][A-Z][A-Z]'),
EmployeeID int NULL, OrderDate datetime NULL
CHECK (OrderDate BETWEEN '01/01/70' AND
    GETDATE()),
RequiredDate datetime NULL, ShipVia int NULL
CHECK (ShipVia IN (1, 2, 3, 4)),
Freight money NULL CHECK (Freight>=0),
ShipCountry nvarchar (15),
CHECK (RequiredDate>OrderDate))
```

Ràng buộc Check

- Ví dụ 4:

```
CREATE TABLE PHANCONG(
        ma_nvien CHAR(9) NOT NULL,
        soda INT NOT NULL,
        thoigian DECIMAL(3,1) NOT NULL,
        PRIMARY KEY (ma_nvien, soda),
        FOREIGN KEY (ma_nvien) REFERENCES
NHANVIEN,
        FOREIGN KEY (soda) REFERENCES
DEAN(mada),
        CHECK (thoigian >= 0))
```

Rule

- Định nghĩa các qui tắc hợp lệ mà có thể kết buộc vào các cột của bảng hay các kiểu dữ liệu do người dùng định nghĩa.
- Rule được tạo nên chính nó trước khi kết buộc vào đối tượng khác
- Định nghĩa Rule:

  CREATE RULE rulename AS condition_expression
- Kết buộc rule vào một cột

  sp_bindrule rulename, tablename.columnname
- Kết buộc Rule vào user-defined datatype

  sp_binrule rulename, datatypename[, futureonly]

Rule

- Ví dụ:

```
CREATE RULE ActiveDate AS
        @Date Between '01/01/70' AND Getdate()
sp_bindrule ActiveDate, 'Orders.OrderDate‘
```

- Chú ý:
  - Futureonly chỉ định các cột tồn tại sẵn mà có dùng kiểu dữ liệu này thì không thể kế thừa Rule mới. Chỉ sử dụng với kiểu dữ liệu, cột thì không.

Các ràng buộc - Constraints

< column_constraint > ::=[CONSTRAINT *constraint_name*]

{ [NULL | NOT NULL]

| [{ PRIMARY KEY | UNIQUE }]

| [[FOREIGN KEY]

REFERENCES *ref_table* [**(** *ref_column* **)**]

[ON DELETE { CASCADE | NO ACTION }]

[ON UPDATE { CASCADE | NO ACTION }]

]

| CHECK **(** *logical_expression* **)**

}

Ràng buộc Primary Key

- SQL Server tự động tạo một chỉ mục cho bảng ứng với các cột tham gia ràng buộc Primary key.
- Mỗi bảng chỉ có thể có duy nhất 1 ràng buộc primary key.
- Ràng buộc Primary key gồm một hay nhiều cột dùng để nhận diện các record, giá trị của primary key không được phép trùng nhau và không chứa giá trị Null.
- Chỉ mục sẽ được tự động tạo ra khi có khai báo 1 ràng buộc primary key.
- Chỉ mục do primary key tạo ra mặc định thường là clustered

Ràng buộc Primary Key

```
CREATE TABLE TableName
    (columname datatype [,…],[CONSTRAINT constraint_name]
    PRIMARY KEY [CLUSTERED|NONCLUSTERED]
    {(column [ASC |DESC][,…,n])}
    [WITH FILLFACTOR = fillfactor]
    [ON {filegroup|DEFAULT}]
```

```
ALTER TABLE TableName
    ADD [CONSTRAINT constraint_name]
    PRIMARY KEY {(column [ASC |DESC][,…,n])}
    [ON {filegroup|DEFAULT}]
```

Ràng buộc Primary Key

- **Ví dụ 1**

```
CREATE TABLE DEAN
(
        MADA smallint PRIMARY KEY
        CLUSTERED NOT NULL,
        TENDA varchar(50) NOT NULL
        DEFAULT 'Chua '
)
```

Ràng buộc Primary Key

Ví dụ 2: Định nghĩa mức cột

```
CREATE TABLE Events (
EventID int NOT NULL PRIMARY KEY,
EventTitle nvarchar (100) NULL ,
EventDescription ntext NULL )
```

Ví dụ 3: Định nghĩa mức bảng

```
CREATE TABLE Sukien (
OrderID int IDENTITY (1, 1) NOT NULL,
CustomerID nchar (5),
PRIMARY KEY NONCLUSTERED (OrderID) WITH
   FILLFACTOR=90 )
```

Ràng buộc Primary Key

Ví dụ 4:

```
CREATE TABLE Table3 (
    col1 int NOT NULL,
    col2 varchar (100) )
```

Thêm ràng buộc khóa chính

```
ALTER TABLE Table3
    ADD CONSTRAINT Table3_PK
    PRIMARY KEY (Col1)
EXEC Sp_helpconstraint Table3
```

Ràng buộc Primary Key

Xóa một Primary key Constraint

```
ALTER TABLE Table3
    DROP CONSTRAINT Table3_PK
```

Lưu ý:

- Không thể xóa một Primary key constraint nếu nó được tham chiếu bởi Foreign key của một bảng khác, muốn xóa phải xóa Foreign key trước

Ràng buộc Unique

- Dùng để đảm bảo không có giá trị trùng ở các cột.
- Một cột hay sự kết hợp giữa các cột vốn không phải là khóa chính.
- Chấp nhận một hàng chứa giá trị Null.
- Một bảng có thể có nhiều Unique constraint.

So sánh Unique và Primary key

- Ràng buộc Primary key gồm một hay nhiều cột dùng để nhận diện các record, giá trị của primary key không được phép trùng nhau và không chứa giá trị Null.
- Ràng buộc UNIQUE được dùng cho các cột không phải là primary key.
- Ràng buộc UNIQUE tương tự như PRIMARY KEY nhưng nó cho phép 1 hàng được quyền có giá trị NULL
- Một bảng có thể có nhiều ràng buộc unique nhưng chỉ có 1 ràng buộc primary key mà thôi.
- Chỉ mục do primary key tạo ra mặc định thường là clustered
- Chỉ mục do unique tạo ra mặc định thường là nonclustered

Ràng buộc Unique

```
CREATE TABLE TableName
    (columname datatype [,…],[CONSTRAINT constraint_name]
    UNIQUE [CLUSTERED|NONCLUSTERED]
    {(column [ASC |DESC][,…,n])}
    [WITH FILLFACTOR = fillfactor]
    [ON {filegroup|DEFAULT}]
```

```
ALTER TABLE TableName
    ADD [CONSTRAINT constraint_name]
    UNIQUE {(column [ASC |DESC][,…,n])}
    [ON {filegroup|DEFAULT}]
```

Ràng buộc Unique

- **Ví dụ 1**

```
CREATE TABLE jobs
(
        job_id smallint UNIQUE
        CLUSTERED NOT NULL,
        job_desc varchar(50) NOT NULL
        DEFAULT 'New Position - title not
        formalized yet'
)
```

Ràng buộc Unique

Ví dụ 2: Định nghĩa mức cột

```
CREATE TABLE Events (
EventID int NOT NULL UNIQUE,
EventTitle nvarchar (100) NULL ,
EventDescription ntext NULL)
```

Ví dụ 3: Định nghĩa mức bảng

```
CREATE TABLE Orders (
OrderID int IDENTITY (1, 1) NOT NULL,
CustomerID nchar (5), UNIQUE NONCLUSTERED
   (OrderID) WITH FILLFACTOR=90 )
```

Ràng buộc Unique

Ví dụ 4:

```
CREATE TABLE Table3Unique (
    col1 int NOT NULL,
    col2 varchar (100) )
```

Thêm ràng buộc khóa chính

```
ALTER TABLE Table3Unique
  ADD col3 char(5) CONSTRAINT Table3_Unique
  UNIQUE
EXEC Sp_helpconstraint Table3
```

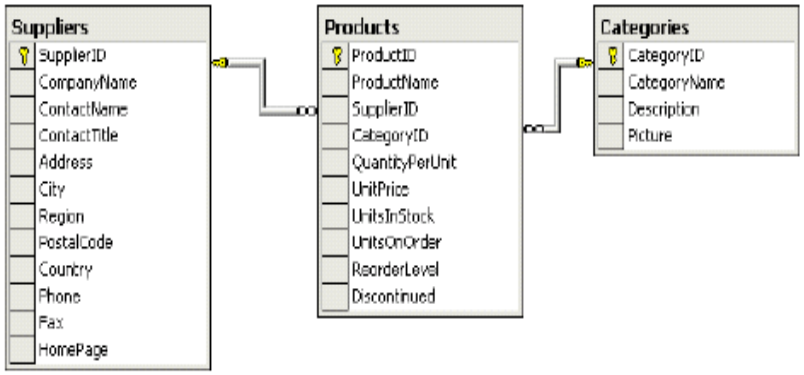
Ràng buộc Unique

Xóa một Unique Constraint

```
ALTER TABLE Table3Unique
    DROP CONSTRAINT Table3_Unique
```

Ràng buộc Foreign key

- Quan hệ chỉ có thể được tạo ra giữa các bảng trong cùng 1 CSDL và trên cùng 1 server.
- Khoá ngoại chỉ có thể tham chiếu đến một cột sau trong bảng chính:
  - Là 1 cột hay 1 phần của khoá chính
  - Là cột có ràng buộc unique
  - Là cột có chỉ mục unique



- Một bảng có thể có tối đa 253 khoá ngoại và có thể tham chiếu đến 253 bảng khác nhau.

Ràng buộc Foreign key

Định nghĩa FOREIGN KEY CONSTRAIT khi tạo bảng

CREATE TABLE TableName

(columnName datatype [,…],

[CONSTRAINT constraintName]

FOREIGN KEY[(column[,..n])]

REFERENCES *ref_table* [**(** **r***ef_column [,..n])]***)**]

[ON DELETE { CASCADE | NO ACTION }]

[ON UPDATE { CASCADE | NO ACTION }]

[NOT FOR REPLICATION]

Ràng buộc Foreign key

ON UPDATE|DELETE {CASCADE | NO ACTION}

- Xác định hành động cần phải thực hiện cho 1 hàng trong bảng đang tạo nếu hàng đó có quan hệ tham chiếu và hàng tham chiếu bị xoá khỏi bảng chính. Mặc định là NO ACTION.
- CASCADE: dùng để xác định là hàng sẽ bị cập nhật/xoá khỏi bảng tham chiếu nếu hàng đó bị cập nhật/xóa khỏi bảng chính
- NO ACTION: SQL Server sẽ đưa ra thông báo lỗi và việc xoá hàng trên bảng chính sẽ bị từ chối.

Ràng buộc Foreign key

- **Ví dụ 1**

```
CREATE TABLE VITRI
(MaVt int Primary key, DiaChi varchar(40))

CREATE TABLE PhongBan
   (        Mapb int primary key,
            TenPb varchar(30),
               MaVT int REFERENCES VITRI(MaVt)
   )
```

Ràng buộc Foreign key

- **Ví dụ 2**

```
CREATE TABLE NHANVIEN (
        manv  CHAR(9) NOT NULL,
        honv VARCHAR(15) NOT NULL,
        tennv VARCHAR(15) NOT NULL,
        ngsinh DATETIME, diachi VARCHAR(30),
        phai CHAR(1), ma_nql CHAR(9),
        phg INT  NOT NULL,
        CONSTRAINT Nv_PK PRIMARY KEY (manv),
        CONSTRAINT Nv_Fk FOREIGN KEY (phg)
        REFERENCES PHONGBAN(mapb))
```

Ràng buộc Foreign key

Định nghĩa FOREIGN KEY CONSTRAINT khi bảng đã tồn tại

ALTER TABLE TableName

[WITH CHECK | WITH NOCHECK] ADD

[CONSTRAINT constraintName]

FOREIGN KEY[(column[,..n])]

REFERENCES *ref_table* [**(r***ef_column [,..n])]***)**]

[ON DELETE { CASCADE | NO ACTION }]

[ON UPDATE { CASCADE | NO ACTION }]

[NOT FOR REPLICATION]

Ràng buộc Foreign key

- WITH CHECK: trước khi tạo ràng buộc, SQL Server sẽ kiểm tra dữ liệu hiện có vi phạm ràng buộc hay không, nếu có sẽ không tạo constraint.
- WITH NOCHECK: tạo constraint mà không cần kiểm tra dữ liệu hiện có có vi phạm ràng buộc hay không.

Ràng buộc Foreign key

- **Ví dụ 1**

```
CREATE TABLE ChucVu
(Macv int primary key, tench varchar(30))

ALTER TABLE NhanVien1
   ADD CV int

ALTER TABLE Nhanvien1
   ADD CONSTRAINT Cv_FK Foreign key (Macv)
   REFERENCES Chucvu(Macv)
)
```

Các mức ràng buộc

- Có thể tạo ràng buộc theo 2 mức :
  - Mức cột (Column level)
  - Mức bảng (Table level)
- Ràng buộc mức bảng:

< table_constraint >::= [CONSTRAINT *constraint_name*]
{ [{ PRIMARY KEY | UNIQUE }[CLUSTERED |
NONCLUSTERED]
{ **(** *column* [ASC | DESC] [**,**...*n*] **)** }]
| FOREIGN KEY [**(** *column* [**,**...*n*] **)**]
REFERENCES *ref_table* [**(** *ref_column* [**,**...*n*] **)**]
[ON DELETE { CASCADE | NO ACTION }]
[ON UPDATE { CASCADE | NO ACTION }]
[NOT FOR REPLICATION]
| CHECK [NOT FOR REPLICATION]
(*search_conditions* **)**}

Các mức ràng buộc

- **Ví dụ về ràng buộc mức bảng**
- Tạo 1 ràng buộc khoá chính ở mức bảng

```
CREATE TABLE cthoadon
(    sohd int NOT NULL,
     MaHang char(4) NOT NULL,
     SoLuong int NOT NULL,
     DonGia money,
     CONSTRAINT Pk_ctHoadon primary key
     clustered (sohd,MaHang)
)
```

Thủ tục lưu trữ hệ thống

sp_help: System stored procedure

- Để kiểm tra xem bảng đã được tạo hay chưa?

sp_help table_name

- Để kiểm tra xem kiểu dữ liệu của người dùng đã được tạo hay chưa?

sp_help datatype_name

Xem Constraints

- **Viewing Constraints**
  - Sp_helpConstraint Events

- **Verify constraints by inserting data**
  - INSERT Events DEFAULT VALUES

	EventID	EventType	EventTitle	EventDescription	EventLanguage
1	1	Party	NULL	NULL	NULL

EventDate	EventEndDate	EventCreator
2004-02-15 01:28:00	2004-02-16 01:28:00	sa

  - SELECT * FROM Events

Bài tập

Example

a) Tạo Table có khóa chính

```
CREATE TABLE KhachHang
(Makh char(5), Tenkh Varchar(40), DiaChi  Varchar(50),
    DienThoai Nvarchar(10) CONSTRAINT Makh_Pk  Primary
    key(Makh))
```

b) Tạo Table có khóa ngoại

```
CREATE TABLE  HoaDon
(Mahd Char(5), NgayLap Datetime, Makh Char(5)  CONSTRAINT Mahd_Pk
    Primary key(Mahd)
    CONSTRAINT Makh_Fk Foreign key References KhachHang (Makh))
```

Modifying Table_Defining Constraints

Example

a) ALTER TABLE Sanpham

ADD CONSTRAINT Masp_pk Primary key(Masp)

b) ALTER TABLE ChiTietHoaDon

ADD CONSTRAINT Masp_Mahd_pk Primary key(Mahd,Masp)

c) ALTER TABLE ChiTietHoaDon

ADD CONSTRAINT Masp_fk Foreign key (Masp) References Sanpham(Masp)

d) ALTER TABLE ChiTietHoaDon

ADD CONSTRAINT Mahd_fk Foreign key(Mahd) References HoaDon(Mahd)

Xóa Constraints

- **Viewing Constraints**

  sp_helpconstraint *tablename*

- **Dropping Constraints**

  ALTER TABLE *tablename*
  DROP [CONSTRAINT] *constraintname*

- **Disabling Constraints**

  ALTER TABLE *tablename*
  NOCHECK CONSTRAINT {ALL | *constraintname* [,...]}

Chỉ mục INDEX

Cơ bản về chỉ mục

- Chỉ mục được tạo ra dựa theo các giá trị được xếp thứ tự từ 1 hay nhiều cột được chọn.
- Chỉ mục được tạo tự động bất cứ lúc nào ta xác định khoá chính hay ràng buộc unique.
- Mỗi bảng chỉ có thể có duy nhất 1 chỉ mục clustered nhưng không bắt buộc là phải có chỉ mục này.
- Một bảng có thể có tới 249 chỉ mục nonclustered

Chỉ mục INDEX

- Nếu bảng không dùng chỉ mục:
  - Các hàng không lưu trữ theo 1 thứ tự đặc biệt nào.
  - Các trang dữ liệu cũng không sắp xếp tuần tự.

Chỉ mục INDEX

Mục đích sử dụng chỉ mục

- Cải thiện việc thực thi khi sắp xếp hay nhóm dữ liệu
- Cải thiện việc thực thi các truy vấn có kết nối giữa các bảng
- Cải thiện tính duy nhất của 1 cột hay nhiều cột
- Mục đích chính của index là để tăng tốc việc truy xuất dữ liệu.
- Hai loại chỉ mục:
  - Clustered index
  - Nonclustered index

Creating Index

```
CREATE [ UNIQUE ] [ CLUSTERED | NONCLUSTERED ] INDEX index_name
    ON <object> ( column [ ASC | DESC ] [ ,...n ] )
    [ INCLUDE ( column_name [ ,...n ] ) ]
    [ WHERE <filter_predicate> ]
    [ WITH ( <relational_index_option> [ ,...n ] ) ]
    [ ON { partition_scheme_name ( column_name )
         | filegroup_name
         | default
         }
    ]
    [ FILESTREAM_ON { filestream_filegroup_name | partition_scheme_name | "NULL" } ]

[ ; ]
```

```
USE library
CREATE CLUSTERED INDEX cl_lastname
ON library..member (lastname)
```

Chỉ mục Clustered và Nonclustered

- Chỉ mục *clustered:*
  - *Dữ liệu được sắp xếp vật lý*.
  - Chỉ có 1 chỉ mục clustered trong mỗi bảng.
  - Tương tự như danh bạ điện thoại (telephone directory) trong đó dữ liệu được sắp xếp bởi tên thuê bao

Ví dụ

```
CREATE CLUSTERED INDEX Customerid_ndx
        ON Orders (Customerid)
```

Chỉ mục Clustered và Nonclustered

- Chỉ mục *nonclustered*: là chỉ mục có cấu trúc riêng biệt độc lập với thứ tự vật lý của bảng dữ liệu.
  - Thứ tự vật lý của chỉ mục nonclustered không trùng với thứ tự các bản ghi trong bảng dữ liệu
  - Tương tự như chỉ mục trong textbook
- Nên tạo chỉ mục clustered trước khi tạo chỉ mục nonclustered để chỉ mục nonclustered không cần phải tạo lại.
- Chỉ mục clustered thực thi nhanh hơn chỉ mục nonclustered.

```
CREATE NONCLUSTERED INDEX Manv_ndx
        ON Nhanvien (Manv)
```

Thuộc tính của Indexes

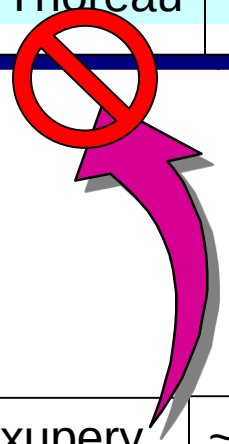
- Chỉ mục Unique: Không cho giá trị trùng nhau trong cột chỉ mục
- Chỉ mục Composite: cho phép hai hay nhiều cột được sử dụng để tạo chỉ mục.

Creating Unique Indexes

```
USE library
CREATE UNIQUE INDEX title_ident
    ON title (title_no)
```

title

title_no	*title*	*author*	*synopsis*
10	The Night-Born	Jack London	~ ~ ~
11	Lemon	Motojirou	~ ~ ~
12	Walking	Henry David Thoreau	~ ~ ~



Duplicate key values are not allowed when a new row is added to the table

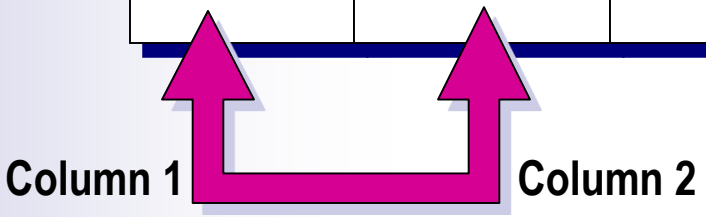
12	Le Petit Prince	Antoine de Saint-Exupery	~ ~ ~

Creating Composite Indexes

```
USE library
CREATE UNIQUE INDEX loan_ident
ON loan (isbn, copy_no)
```

loan

isbn	*copy_no*	*title_no*	*member_no*	*out_date*
342	5	35	3744	1998-01-06
342	10	35	5278	1998-01-04
343	4	35	3445	1998-01-04



Composite Key

Xem - Xóa chỉ mục

✓Xem chỉ mục

Cú pháp

sp_helpindex <table_name>

✓Xóa chỉ mục

Cú pháp

DROP INDEX <table_name>.<index_name>