



CSDL NC Seminar I

Tiết Gia Hồng

 **Bộ môn HTTT – Khoa CNTT – ĐH KHTN**

Stored Procedure

- ⑩ Stored Procedure là thủ tục được biên dịch và lưu trữ sẵn trong CSDL

```
CREATE PROCEDURE XinChao
    @hoTen nvarchar(50)
AS
    print N'Xin chào ' + @hoTen
GO
```

- ⑩ Lợi ích:

- ☞ Tăng hiệu năng
- ☞ Độc lập với chương trình ứng dụng
- ☞ Giảm vấn đề nghẽn đường truyền mạng (client-server)
- ☞ Bảo mật cơ sở dữ liệu

Tạo Stored Procedure bằng lệnh T-SQL

```
CREATE PROCEDURE procedure_name
```

Tạo stored
procedure

```
@parameter1 data_type    [output],  
@parameter2 data_type    [output],...
```

Khai báo tham số
(input, output)

```
AS
```

```
[khai báo các biến cho xử lý]  
{Các câu lệnh transact-sql}
```

```
GO
```

Xử lý stored
procedure

Thực thi Stored Procedure

```
EXECUTE procedure_name      parameter_value1,  
parameter_value2,...
```

```
EXEC    procedure_name      parameter_value1,  
parameter_value2,...
```

```
CREATE PROCEDURE    XinChao  
    @hoTen nvarchar(50)  
AS  
    print N'Xin chào ' + @hoTen  
GO
```

```
EXEC XinChao N'Hiệp'
```

Cập nhật & Xóa Stored Procedure

10 Cập nhật stored procedure

```
ALTER PROCEDURE  procedure_name
    @parameter1 data_type      [output],
    @parameter2 data_type      [output],... /*các tham số*/
AS
    [khai báo các biến cho xử lý]
    {Các câu lệnh transact-sql}
GO
```

```
DROP PROCEDURE  procedure_name
```

```
DROP PROC       procedure_name
```

Trả về kết quả từ Stored Procedure

- ⑩ Trả về giá trị bằng tham số **output**
- ⑩ Trả về giá trị bằng lệnh **return**
- ⑩ Trả về bảng dữ liệu bằng lệnh **select**

Trả về kết quả bằng tham số output

```
ALTER PROC Tru
    @So1    int,
    @So2    int,
    @Kq     int output
AS
    set @Kq = @So1 - @So2
GO

DECLARE @test int
EXEC Tru 1, 2, @test output
PRINT @test
```

Trả về giá trị bằng lệnh return

```
CREATE PROC Test
    @Lenh    int
AS
    if (@Lenh = 1)
        return 1 1

    if (@Lenh = 2)
    begin
        declare @float float
        set @float = 2.6


        return @float 2
    end

    if (@Lenh = 3)
    begin
        declare @char varchar(50)
        set @char = 'hello'

        return @char Báo lỗi
    end
end

GO
```

```
declare @test float
EXEC @test = Test 3
print @test
```



Chỉ trả về được số
nguyên

Trả về giá trị bằng giá trị bằng lệnh select

```
CREATE PROC TestSelect
AS
    SELECT * FROM SINHVIEN

GO

EXEC TestSelect
```

Results

Messages

	ma	hoTen	namSinh	danT...	maLop
1	0212001	Nguyễn Vĩnh An	1984	Kinh	TH2002/01
2	0212002	Nguyễn Thanh Bình	1985	Kinh	TH2002/01
3	0212003	Nguyễn Thanh Cường	1984	Kinh	TH2002/02
4	0212004	Nguyễn Quốc Duy	1983	Kinh	TH2002/02
5	0312001	Phan Tuấn Anh	1985	Kinh	VL2003/01
6	0312002	Huỳnh Thanh Sang	1984	Kinh	VL2003/01

	ma	maKhoaHoc	maKhoa	maChuongTrinh	soThuTu
1	TH2002/01	K2002	CNTT	CQ	1
2	TH2002/02	K2002	CNTT	CQ	2
3	VL2003/01	K2003	VL	CQ	1

✓

Query executed successfully.

127.0.0.1 (9.0 RTM)

sa (51)

QLSV

00:00:00

9 rows

Thủ tục lồng nhau

```
Create proc A  
AS  
Begin  
    -- Các lệnh  
End
```

```
Create proc B  
AS  
Begin  
    EXEC A  
    -- Các lệnh  
End
```

Ví dụ

--Khai báo kiểu dữ liệu mới

```
CREATE TYPE DSCTDonHang AS TABLE  
(  
    MaSP int UNIQUE,  
    DonGia float,  
    SoLuong int  
)
```

--Ví dụ thêm dữ liệu vào bảng @temp

```
DECLARE @temp DSCTDonHang  
INSERT @temp VALUES(1,15,3)  
SELECT * FROM @temp
```

Ví dụ (tt)

```
CREATE PROC USP_THEMPDH
    @TEMP AS DSCTDATHANG readonly,
    @MADATHANG int
    @MAKHACHHANG int

AS
BEGIN
    --Thêm phiếu đặt hàng
    INSERT PHIEUDATHANG (MADATHANG, NGAYDAT, MAKHACHHANG)
    VALUES(@MADATHANG, GETDATE(), @MAKHACHHANG)

    --Thêm chi tiết phiếu đặt hàng
    INSERT CHITIETPHIEUDAT ( MASANPHAM, DONGIA, SOLUONG,
    MADATHANG)
    SELECT *, @MADATHANG FROM @TEMP
END
```

Ví dụ (tt)

--Khai báo danh sách chi tiết đơn hàng

```
DECLARE @TEMP DSCTDATHANG
```

--Thêm chi tiết vào danh sách

```
INSERT @TEMP  
VALUES(1, 10, 2),  
      (2, 20, 3)
```

--Xem nội dung bảng @temp

```
SELECT * FROM @TEMP
```

--Thực thi thủ tục

```
EXEC USP_THEMPDH @TEMP, 'DH001', 'KH1'
```

Function

- ⑩ Function là hàm được biên dịch và lưu trữ sẵn trong cơ sở dữ liệu
- ⑩ Phân loại function:
 - ⌘ Function trả về giá trị vô hướng (Scalar Function)
 - ⌘ Function trả về bảng dữ liệu

Function trả về giá trị vô hướng

```
CREATE FUNCTION Cong  
    (@so1 int, @so2  
     int)  
RETURNS int  
AS  
BEGIN  
    RETURN @so1 + @so2  
END  
GO
```

```
SELECT dbo.Cong(1, 2)
```

Bắt buộc phải có
BEGIN...END

Khi gọi Scalar Function
phải sử dụng thêm tên
schema

Có thể kết hợp function
trong biểu thức

Function đóng vai trò view

```
CREATE FUNCTION TimNhanVien  
    (@MaPhong      char(5))  
RETURNS table  
AS  
BEGIN ERROR  
RETURN (SELECT nv.MaNV, nv.HoTen  
          FROM NhanVien nv  
          WHERE nv.MaPhong = @MaPhong)  
END  
GO
```

```
SELECT *  
FROM TimNhanVien('PB001')
```

Không có BEGIN...END
trong inline table-
valued function

Có thể gọi function
trong lệnh FROM

Không bắt buộc
có tên schema

Partition - Nhắc lại

- Sau khi **thiết kế ER**, tinh chế lược đồ, và định nghĩa các khung nhìn, chúng ta có **lược đồ ở mức quan niệm**.
- Bước tiếp theo là giai đoạn **thiết kế logic**. Đây là bước trung gian để giai đoạn thiết kế vật lý được dễ dàng.
- Giai đoạn **thiết kế vật lý** là lựa chọn cài đặt các chỉ mục, tinh chỉnh lại lược đồ quan niệm để đáp ứng được mục tiêu hiệu suất.

Giới thiệu

Một số vấn đề khi thiết kế dữ liệu mức vật lý

- ❖ Thiết kế field
- ❖ Phân chia dữ liệu (partition)
- ❖ Gộp dữ liệu (denormalization)
- ❖ Tổ chức file chỉ mục

Thiết kế dữ liệu vật lý

- ❖ Phân mảnh dữ liệu là gì?
- ❖ Tại sao phải phân mảnh?

Thiết kế dữ liệu vật lý

❖ Phân mảnh dữ liệu (partition)

- Cải thiện khả năng co giãn và khả năng quản lý các bảng lớn (large table)
- Khi các bảng và các chỉ mục quá lớn, việc phân mảnh giúp chia dữ liệu thành các phần nhỏ hơn, có thể quản lý được.
- Nếu 1 bảng lớn tồn tại trên hệ thống nhiều CPUs, việc phân mảnh bảng sẽ giúp tăng hiệu suất khi thực hiện song song

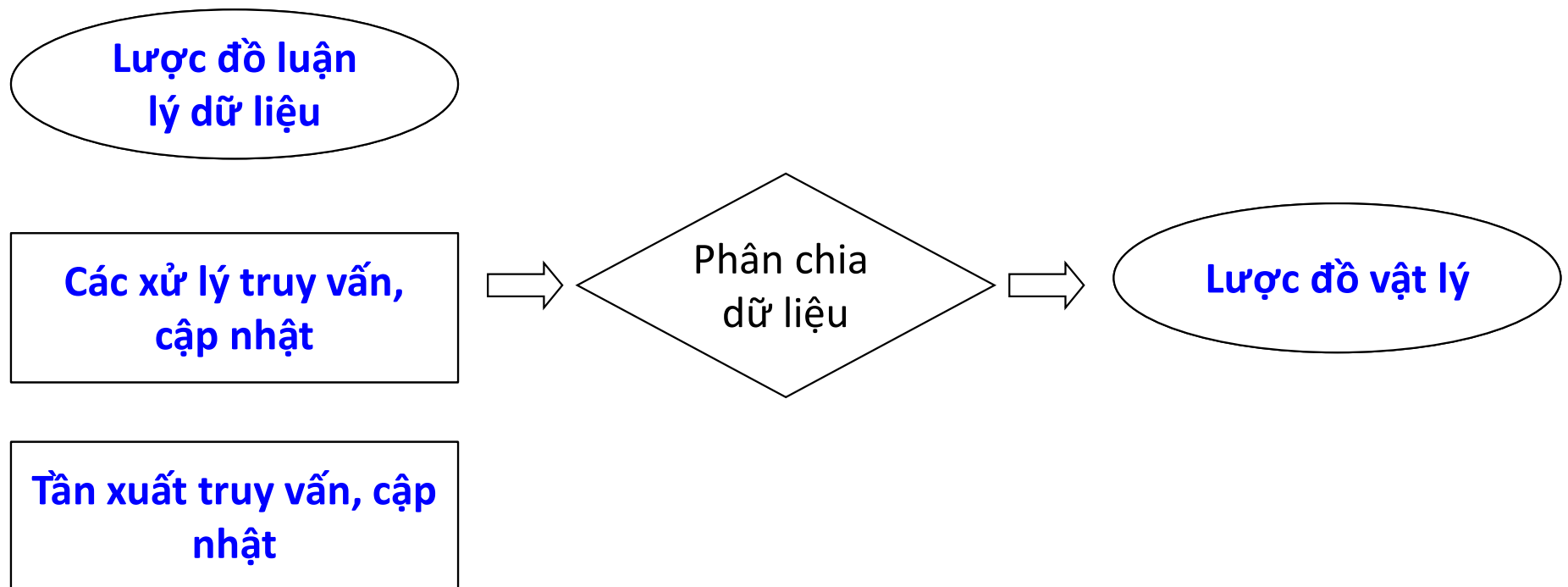
Thiết kế dữ liệu vật lý

❖ Phân chia dữ liệu (partition)

- **Phân chia theo chiều ngang** (horizontal partition): chia bảng dữ liệu thành nhiều bảng cùng số cột
- **Tình huống áp dụng**: khi nhiều người dùng khác nhau cần truy cập các dòng dữ liệu khác nhau
- **Ưu điểm**:
 - Tối ưu hóa tốc độ truy cập dữ liệu
- **Nhược điểm**
 - Phức tạp khi phải truy cập toàn bộ dữ liệu

Thiết kế dữ liệu vật lý

❖ Phân chia dữ liệu (partition)



Thiết kế dữ liệu vật lý

❖ Phân chia dữ liệu (partition)

■ Ví dụ:

KL: ~10.000.000/năm

HOA_DON

Số_HD	Ngày_HD	Diễn_giải	Trị giá
Hd00001	1/1/04	Xxxxxxx	1.000.000
Hd00002	2/1/04	Yyyyyyy	2.000.000
....			
Hd15000	1/1/05	Zxzxzzxzx	1.400.000
Hd15001	2/1/05	Qqqqqqqq	2.100.000
...			
Hd30000	2/1/06	Asasasas	12.000.000
Hd30001	2/1/06	Dsdsdsds	1.000.000

Các xử lý truy cập dữ liệu

Mã số	Tên xử lý	Tần suất
O1	Tìm hóa đơn	100/ngày
O2	Tính doanh thu tháng	1/tháng
O3	Tính doanh thu theo khách hàng	100/tháng
O4	Tổng hợp doanh số năm	1/năm
O5	Lập biểu đồ so sánh doanh số theo các năm	1/năm

Thiết kế dữ liệu vật lý

❖ Thực hiện phân mảnh ngang:

- Tạo các **filegroup** để chứa các phân mảnh
- Tạo **partition function** (hàm phân mảnh) ánh xạ các dòng của bảng hoặc các chỉ mục trong phân mảnh dựa vào tiêu chí phân mảnh
- Tạo lược đồ **partition scheme** ánh xạ các phân mảnh của bảng vào các filegroup

Hướng dẫn tạo partition

❖ Bước 1: tạo các filegroup

```
CREATE DATABASE DBForPartitioning  
ON  
PRIMARY
```

```
(NAME='DBForPartitioning_1',  
FILENAME= 'D:\PartitionDB\FG1\DBForPartitioning_1.mdf',  
SIZE=2,  
MAXSIZE=100,  
FILEGROWTH=1 ),
```

FILEGROUP FG2

```
(NAME = 'DBForPartitioning_2',  
FILENAME = 'D:\PartitionDB\FG2\DBForPartitioning_2.ndf', SIZE = 2, MAXSIZE=100,  
FILEGROWTH=1 ),
```

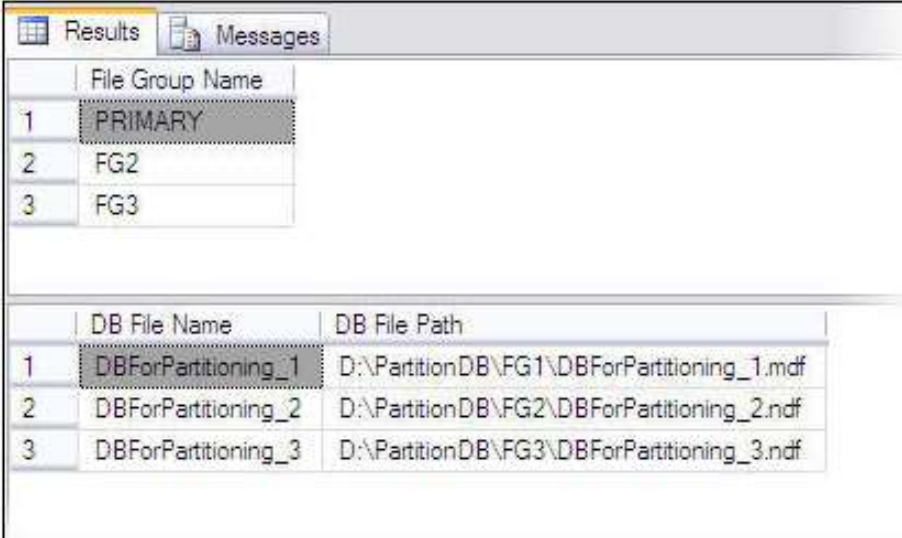
FILEGROUP FG3

```
(NAME = 'DBForPartitioning_3',  
FILENAME ='D:\PartitionDB\FG3\DBForPartitioning_3.ndf',  
SIZE = 2,  
MAXSIZE=100,  
FILEGROWTH=1 )  
GO
```

Hướng dẫn tạo partition

❖ Xem lại các filegroup đã tạo

```
Use DBForPartitioning
GO
-- Confirm Filegroups
SELECT name as [File Group Name]
FROM sys.filegroups
WHERE type = 'FG'
GO
```



	File Group Name
1	PRIMARY
2	FG2
3	FG3

	DB File Name	DB File Path
1	DBForPartitioning_1	D:\PartitionDB\FG1\DBForPartitioning_1.mdf
2	DBForPartitioning_2	D:\PartitionDB\FG2\DBForPartitioning_2.ndf
3	DBForPartitioning_3	D:\PartitionDB\FG3\DBForPartitioning_3.ndf

```
-- Confirm Datafiles
SELECT name as [DB FileName], physical_name as
[DB File Path]
FROM sys.database_files
where type_desc = 'ROWS'
GO
```

Hướng dẫn tạo partition

❖ Bước 2: tạo partition function

```
Use DBForPartitioning
GO
CREATE PARTITION FUNCTION salesYearPartitions (datetime)
AS
    RANGE RIGHT
    FOR VALUES ( '2009-01-01', '2010-01-01')
GO
```

Hướng dẫn tạo partition

❖ Bước 3: tạo partition schema & partitioned table

```
Use DBForPartitioning
GO
CREATE PARTITION SCHEME Test_PartitionScheme
AS
    PARTITION salesYearPartitions
    TO ([PRIMARY], FG2, FG3 )
GO
```

```
Use DBForPartitioning
GO
CREATE TABLE SalesArchival
(
    SaleTime datetime PRIMARY KEY,
    ItemName varchar(50)
)
ON Test_PartitionScheme (SaleTime);
GO
```

Hướng dẫn tạo partition

```
SELECT p.partition_number AS PartitionNumber,  
       f.name AS PartitionFilegroup,  
       p.rows AS NumberOfRows  
FROM sys.partitions p JOIN sys.destination_data_spaces dds  
ON p.partition_number = dds.destination_id  
JOIN sys.filegroups f ON dds.data_space_id = f.data_space_id  
WHERE OBJECT_NAME(OBJECT_ID) = 'SalesArchival'
```

```
INSERT SalesArchival  
VALUES('2009-1-12', '1'),  
      ('2010-12-1', '2'),  
      ('2009-11-12', '1'),  
      ('2009-2-1', '2'),  
      ('2010-1-12', '1'),  
      ('2009-1-1', '2'),  
      ('2008-1-12', '1'),  
      ('2008-12-1', '2')
```

	PartitionNumber	PartitionFilegroup	NumberOfRows
1	1	PRIMARY	0
2	2	FG2	0
3	3	FG3	0

	PartitionNumber	PartitionFilegroup	NumberOfRows
1	1	PRIMARY	2
2	2	FG2	4
3	3	FG3	2

B

Hướng dẫn tạo partition – ExistTab

❖ Thực hiện phân mảnh ngang:

- Tạo các **filegroup** để chứa các phân mảnh
- Tạo **partition function** (hàm phân mảnh) ánh xạ các dòng của bảng hoặc các chỉ mục trong phân mảnh dựa vào tiêu chí phân mảnh
- Tạo lược đồ **partition scheme** ánh xạ các phân mảnh của bảng vào các filegroup
- Tạo **clustered index** cho bảng dựa trên partition scheme

Hướng dẫn tạo partition – ExistTab

❖ Bước 1: Tạo filegroup

USE QL_PHIM

GO

ALTER DATABASE QL_PHIM

ADD FILEGROUP FG4

ALTER DATABASE QL_PHIM

ADD FILEGROUP FG5

ALTER DATABASE QL_PHIM

ADD FILE (NAME = FG4_2000,

FILENAME = 'E:\...\DBPartition_4.

SIZE = 1MB,

MAXSIZE = UNLIMITED,

FILEGROWTH = 1

) TO FILEGROUP FG4

ALTER DATABASE QL_PHIM

ADD FILE (NAME = FG5_2001,

FILENAME = 'E:\...\DBPartition_5.

SIZE = 1MB,

MAXSIZE = UNLIMITED,

FILEGROWTH = 1

) TO FILEGROUP FG5

	File Group Name
1	PRIMARY
2	FG4
3	FG5

	DB FileName	DB File Path
1	QL_PHIM	C:\Program Files\Microsoft SQL Server\MSSQL13.MSS...
2	FG4_2000	E:\PartitionDB\FG4\DBPartition_4.ndf
3	FG5_2001	E:\PartitionDB\FG4\DBPartition_5.ndf

Hướng dẫn tạo partition – ExistTab

❖ Bước 2, 3: Tạo partition function & scheme

```
CREATE PARTITION FUNCTION salesYearPartitions(DATE)
AS RANGE LEFT
FOR VALUES( '2000-12-31' , '2001-12-31' )
--
CREATE PARTITION SCHEME salesYearPartitionsScheme
AS PARTITION salesYearPartitions
TO (FG4,FG5,[PRIMARY])
```


Hướng dẫn tạo partition – ExistTab

❖ Bước 4: Tạo clustered index trên cột chia

--Xóa khóa chính (nếu có)

```
ALTER TABLE MUONPHIM  
DROP CONSTRAINT PK_MP
```

--Tạo khóa chính với non-clusterIndex

```
ALTER TABLE MUONPHIM  
ADD PRIMARY KEY  
NONCLUSTERED(CMND,TENPHIM, STT ASC)  
ON [PRIMARY]
```

--Tạo clusterIndex cho thuộc tính partition

```
CREATE CLUSTERED INDEX IX_NGAYMUON_DATE  
ON MUONPHIM  
(  
    NGAYMUON  
) ON salesYearPartitionsScheme(NGAYMUON)
```

Hướng dẫn tạo partition – ExistTab

```
SELECT p.partition_number AS partition_number,  
       f.name AS file_group,  
       p.rows AS row_count  
FROM sys.partitions p JOIN sys.destination_data_spaces dds ON  
p.partition_number = dds.destination_id  
JOIN sys.filegroups f ON dds.data_space_id = f.data_space_id  
WHERE OBJECT_NAME(OBJECT_ID) = 'MUONPHIM'  
order by partition_number;
```

	CMND	TENPHIM	STT	NGAYMUON
1	111	AA	1	2000-12-30
2	1111	AA	1	2000-12-30
3	11111	AA	1	2000-12-30
4	121	ab	2	2001-12-01

	partition_number	file_group	row_count
1	1	FG4	3
2	2	FG5	1
3	3	PRIMARY	0

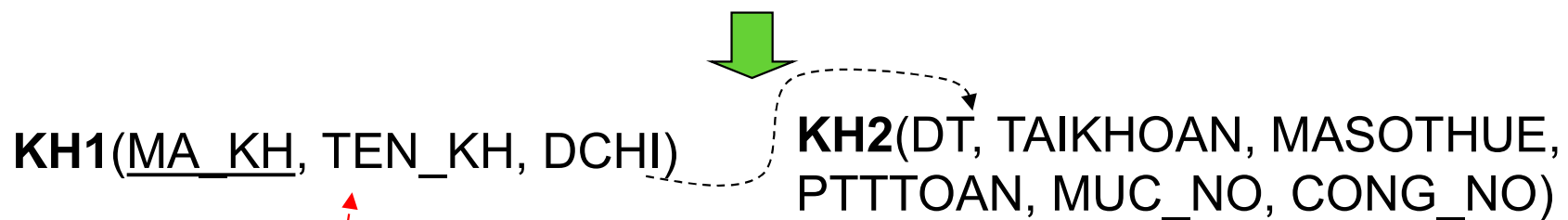
Thiết kế dữ liệu vật lý

❖ Phân chia dữ liệu (partition)

■ **Phân chia theo chiều dọc** (vertical partition):

- Phân chia một cấu trúc luận lý thành những cấu trúc lưu trữ vật lý khác nhau
- Ví dụ:

KHÁCH_HANG(MÃ_KH, TÊN_KH, DCHI, DT, TAIKHOAN, MASOTHUE, PTTTOAN, MUC_NO, CONG_NO)



Cấu trúc truy cập thường xuyên

Cấu trúc truy cập không thường xuyên

Thiết kế dữ liệu vật lý

❖ Gộp dữ liệu (denormalization)

- Mục tiêu:
 - Tối ưu hóa truy vấn dữ liệu
- Hạn chế:
 - Phát sinh trùng lặp dữ liệu
 - Kiểm soát tính nhất quán dữ liệu

Thiết kế dữ liệu vật lý

❖ Gộp dữ liệu (denormalization)

- Gộp 2 quan hệ liên kết 1-1

SINH_VIÊN(MÃ_SV, TEN_SV, CHUYEN_NGANH, NGÀY_SINH)

HỒSƠ_HBÔNG(MÃ_HS, NGÀY_HS, KHẢ_NĂNG, MA_SV)

Xử lý	Dữ liệu liên quan
O1	MÃ_SV, TEN_SV, CHUYEN_NGANH, NGÀY_SINH, KHẢ_NĂNG
O2	TEN_SV, NGÀY_HS, KHẢ_NĂNG

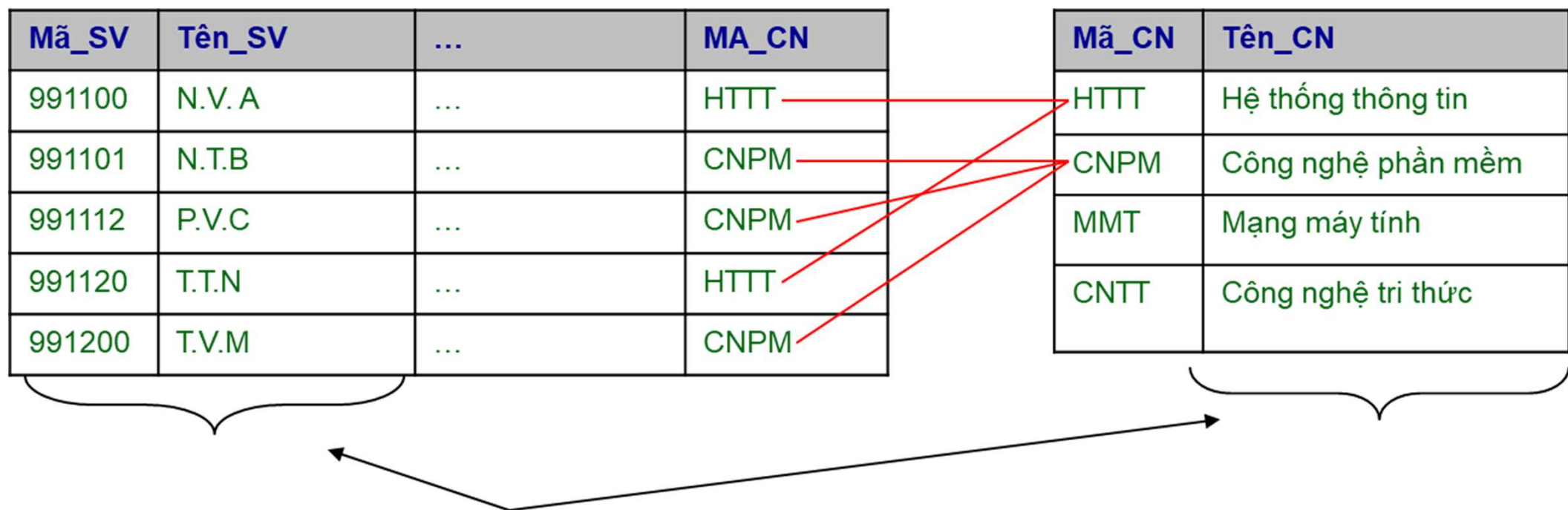


SINH_VIÊN(MÃ_SV, TEN_SV, CHUYEN_NGANH, NGÀY_HS, KHẢ_NĂNG, NGÀY_SINH)

Thiết kế dữ liệu vật lý

❖ Gộp dữ liệu (denormalization)

- Gộp 2 quan hệ liên kết 1-N



Thiết kế dữ liệu vật lý

❖ Gộp dữ liệu (denormalization)

- Gộp 2 quan hệ liên kết 1-N

Mã_SV	Tên_SV	...	MA_CN	Tên_CN
991100	N.V.A	...	HTTT	Hệ thống thông tin
991101	N.T.B	...	CNPM	Công nghệ phần mềm
991112	P.V.C	...	CNPM	Công nghệ phần mềm
991120	T.T.N	...	HTTT	Hệ thống thông tin
991200	T.V.M	...	CNPM	Công nghệ phần mềm

Truy vấn thường xuyên:

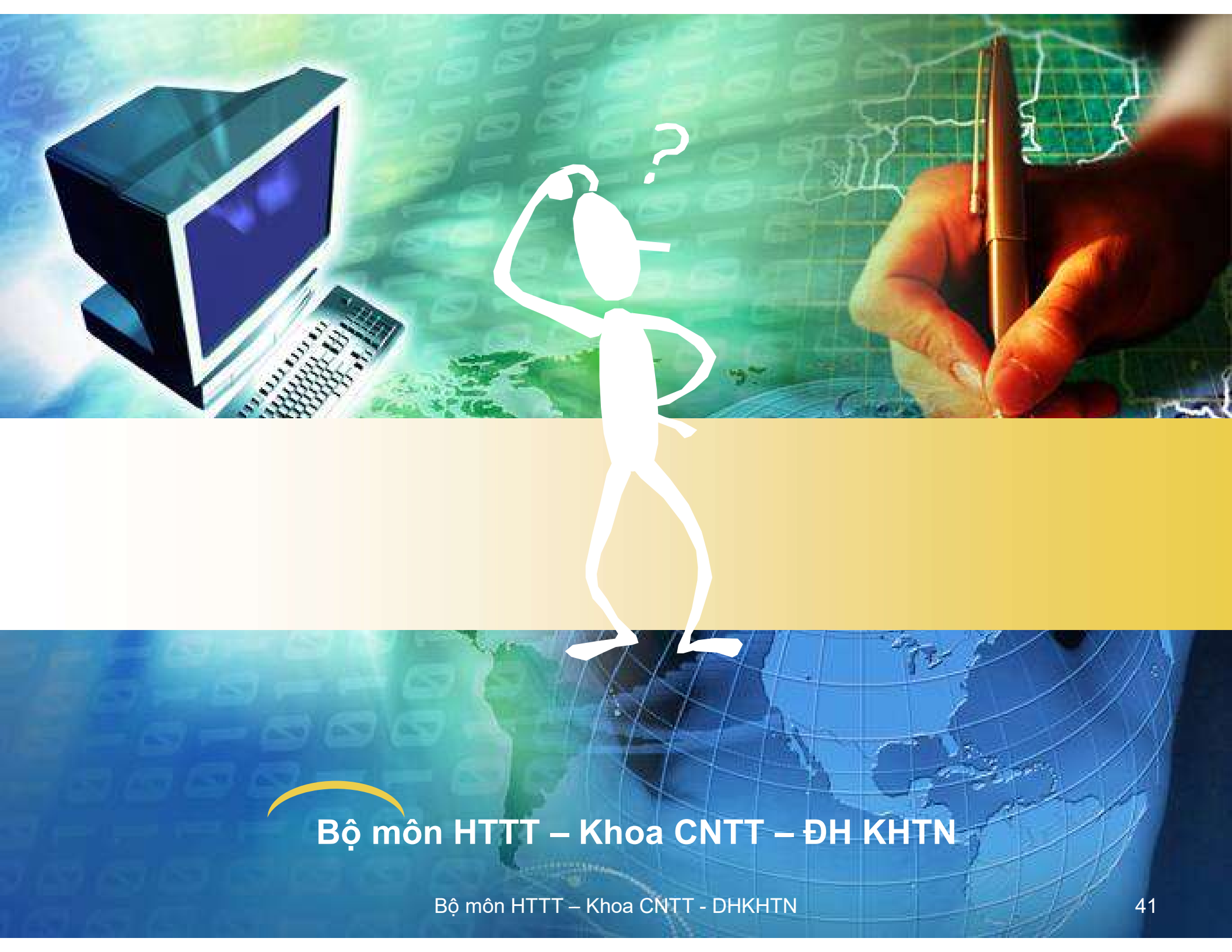
- Q1 (Mã_SV, TÊN_SV, TÊN_CN)

Trùng lặp thông tin

Cấu trúc gộp trên sẽ tối ưu hơn cho truy vấn Q1, nhưng sẽ dẫn đến trùng lặp thông tin

Generate test data

- ❖ <https://www.guru99.com/test-data-generation-tools.html>
- ❖ <https://www.mockaroo.com/>
- ❖ <https://generatedata.com/>



 Bộ môn HTTT – Khoa CNTT – ĐH KHTN