

Database Management System

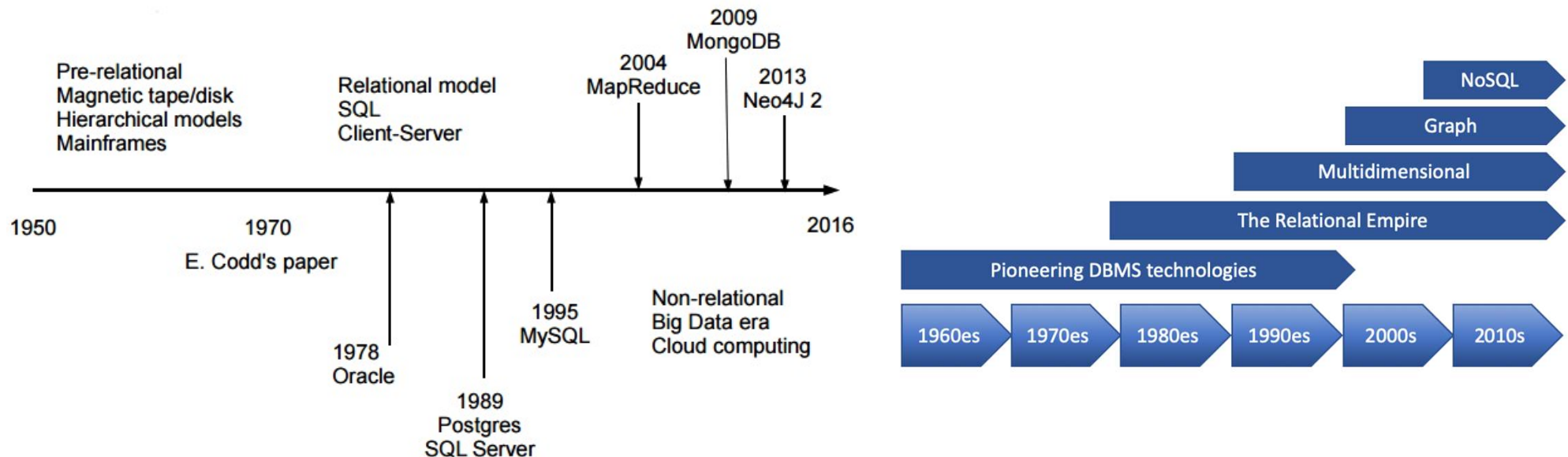
Advanced SQL Commands P2

Dr. Tran Anh Tuan,
Faculty of Mathematics and Computer Science,
University of Science, HCMC

Dr. Tran Anh Tuan, Faculty of Mathematics and Computer
Science, University of Science, HCMC

The database evolution happened in five “waves”:

- The first wave consisted of network, hierarchical, inverted list, and (in the 1990's) object-oriented DBMSs; it took place from roughly 1960 to 1999.
- The relational wave introduced all of the SQL products (and a few non-SQL) around 1990 and began to lose users around 2008.
- The decision support wave introduced Online Analytical Processing (OLAP) and specialized DBMSs around 1990, and is still in full force today.
- The graph wave began with The Semantic Web stack from the Worldwide Web Consortium in 1999, with property graphs appearing around 2008
- The NoSQL wave includes big data and much more; it began in 2008.



Dr. Tran Anh Tuan, Faculty of Mathematics and Computer
Science, University of Science, HCMC

Syllabus

- Lecture 1 : Basic SQL Commands
- **Lecture 2 : Advanced SQL Commands**
- Lecture 3 : Stored Procedures and Functions
- Lecture 4 : Trigger, Transaction, Cursor and Temporary Table
- Lecture 5 : Security, Role, Server Backup, and Server Recovery
- Lecture 6 : Data Synchronization, SQL Profiler, Linked Server, Job Schedule
- Lecture 7 : Basic MongoDB Commands
- Lecture 8 : Advanced MongoDB Commands
- Lecture 9 : SQL Server and MongoDB connection from Application
- Lecture 10 : Final Project

SUM ... OVER



- OVER cho phép bạn lấy được thông tin “kết/tổng” mà không cần dùng GROUP BY. Nói cách khác là bạn vẫn có thể truy xuất tất cả các dòng kèm theo thông tin tổng theo nhóm

```
SELECT SUM(Cost) OVER (PARTITION BY CustomerNo) AS Cost  
      , OrderNum, CustomerNo  
FROM Orders
```

Cost	OrderNum	CustomerNo
8.00	345	1
8.00	346	1
8.00	347	1
2.00	348	2

SUM ... OVER



- Giả sử bảng Order ta có như sau

```
select customerID, productID, orderDate, orderAmount
from Orders
```

customerID	productID	orderDate	orderAmount
1	1	2007-01-01 00:00:00.000	20.00
1	2	2007-01-02 00:00:00.000	30.00
1	2	2007-01-05 00:00:00.000	23.00
1	3	2007-01-04 00:00:00.000	18.00
2	1	2007-01-03 00:00:00.000	74.00
2	1	2007-01-06 00:00:00.000	34.00
2	2	2007-01-08 00:00:00.000	10.00

(7 row(s) affected)

- Kết quả truy vấn vẫn đảm bảo đủ số dòng nhưng kèm thêm thông tin tổng theo nhóm

```
select customerID, productID, orderDate, orderAmount,
       sum(orderAmount) OVER (Partition by CustomerID) as Total
from Orders
```

customerID	productID	orderDate	orderAmount	Total
1	1	2007-01-01 00:00:00.000	20.00	91.00
1	2	2007-01-02 00:00:00.000	30.00	91.00
1	2	2007-01-05 00:00:00.000	23.00	91.00
1	3	2007-01-04 00:00:00.000	18.00	91.00
2	1	2007-01-03 00:00:00.000	74.00	118.00
2	1	2007-01-06 00:00:00.000	34.00	118.00
2	2	2007-01-08 00:00:00.000	10.00	118.00

(7 row(s) affected)

SUM ... OVER



- Với cách này ta dễ dàng tính được Tỷ Lệ theo từng dòng

```
select customerID, productID, orderDate, orderAmount,  
       orderAmount / sum(orderAmount) OVER (Partition by CustomerID) as Pct  
from Orders
```

customerID	productID	orderDate	orderAmount	Pct
1	1	2007-01-01 00:00:00.000	20.00	0.2197
1	2	2007-01-02 00:00:00.000	30.00	0.3296
1	2	2007-01-05 00:00:00.000	23.00	0.2527
1	3	2007-01-04 00:00:00.000	18.00	0.1978
2	1	2007-01-03 00:00:00.000	74.00	0.6271
2	1	2007-01-06 00:00:00.000	34.00	0.2881
2	2	2007-01-08 00:00:00.000	10.00	0.0847

(7 row(s) affected)

AVG ... OVER



- Ngoài ra ta có thể dùng AVG kết hợp với OVER (tương tự)

```
select x.*  
from  
(  
    select customerID, productID, orderDate, orderAmount,  
           avg(orderAmount) over (partition by productID) as ProductAvg  
    from orders  
) x  
where x.orderAmount < x.productAvg
```

customerID	productID	orderDate	orderAmount	ProductAvg
1	1	2007-01-01 00:00:00.000	20.00	42.6666
2	1	2007-01-06 00:00:00.000	34.00	42.6666
2	2	2007-01-08 00:00:00.000	10.00	21.00

(3 row(s) affected)

Các hàm dùng cho Date



Function	Mô tả
GETDATE()	Trả về ngày tháng và thời gian hiện tại
DATEPART()	Trả về một phần của ngày tháng/thời gian
DATEADD()	Thêm hoặc bớt một khoảng thời gian nhất định từ một ngày
DATEDIFF()	Trả về thời gian giữa hai ngày tháng
CONVERT()	Hiển thị dữ liệu ngày tháng/thời gian trong các định dạng khác nhau

Các hàm dùng cho Date



- **Lấy ngày của hệ thống**
 - - Ví Dụ: `Select getdate() as Ngay`
- **Trả về giá trị của ngày tháng năm**
 - - Ví Du: Trả về giá trị của năm
`select datepart (year,'2009/3/15') as nam`
Trả về giá trị của ngày
`select datepart (day,'2009/3/15') as ngay`
- **Trả về giá trị ngày định dạng tương ứng**
 - Ví Dụ: `select datename (dw , '01/06/2000')`
Kết quả: Thursday
-

Các hàm dùng cho Date



- **Hàm trừ thời gian**

- - Ví dụ : `select DATEDIFF (day,'01/06/2000','01/10/2000')`

Kết quả: 4

- `select DATEDIFF (month,'01/06/2000','06/10/2000')`

Kết quả: 5

- **Lấy ngày hiện tại**

- - Ví dụ : `select getdate()`

- **Chữ viết tắt và giá trị**

- Hour = hh (0 - 23) , Minute = Mi (0 - 59) , Second = Ss (0 - 59) ,
- Millisecond = Ms (0 - 999) , Day of year = Dy (1 - 366) ,
- Day = Dd (1 - 31) , Year = yy (1753 - 9999) .

Các hàm dùng cho Date



- **Giá trị ngày cộng với một số**
 - - Ví dụ: `select dateadd (mm, 4 , '01/01/99')`
 - Thêm vào trường mm (tháng) với giá trị 4 . Kết quả sẽ là : 05/01/99
- **Giá trị chênh lệch giữa hai ngày**
 - - Ví dụ: `select datediff (mm , '01/01/99' , '05/01/99')`
 - Trả về kết quả là : 4 . So sánh giữa hai trường tháng .
- **Giá trị ngày định dạng tương ứng**
 - - Ví dụ: `select datename (dw , ' 01/01/2000 ')`
 - Kết quả sẽ trả về Saturday.

COALESCE



- - SQL hỗ trợ chức năng COALESCE để trả về biểu thức có giá trị khác NULL đầu tiên trong số những biểu thức được truyền vào COALESCE.
- - Ví dụ : ta có bảng dữ liệu sau

Table *Contact_Info*

Name	Business_Phone	Cell_Phone	Home_Phone
Jeff	531-2531	622-7813	565-9901
Laura	NULL	772-5588	312-4088
Peter	NULL	NULL	594-7477

- - Ta muốn xuất dữ liệu gồm Name và Contact Phone (chọn mức độ ưu tiên Phone từ trái qua phải) thì làm sao ?

COALESCE



- Ta dùng câu lệnh sau

```
SELECT  
    Name,  
    COALESCE (Business_Phone, Cell_Phone, Home_Phone) AS Contact_Phone  
FROM Contact_Info;
```

Result:

<u>Name</u>	<u>Contact_Phone</u>
Jeff	531-2531
Laura	772-5588
Peter	594-7477

Hàm hệ thống



db_id (database_name) : Mã số định danh cơ sở dữ liệu

db_name (database_id) : Tên cơ sở dữ liệu

host_id() : Số định danh của trạm làm việc

host_name() : Tên trạm làm việc

isnull (expr , value) : Giá trị của biểu thức (expr) sẽ được thay thế với giá trị này.

object_id (' object_name ') : Số định danh của đối tượng cơ sở dữ liệu.

object_name (' object_id ') : Tên của đối tượng cơ sở dữ liệu

suser_sid (' login_name ') : Số định danh bảo mật sid đối với tên đăng nhập của người sử dụng

suser_id (' login_name ') : Số định danh đăng nhập người dùng

suser_sname (server_user_id) : Trả về tên định danh đăng nhập từ số định danh bảo mật của người sử dụng

suser_name (server_user_id) : Tên định danh đăng nhập của người sử dụng

user_id (user_name) : Số định danh cơ sở dữ liệu của người sử dụng

user_name (user_id) : Tên cơ sở dữ liệu của người sử dụng.

Hàm hệ thống



```
SELECT DB_ID('Northwind')
SELECT DB_NAME(5)
SELECT HOST_ID()
SELECT HOST_NAME()

SELECT *, ISNULL(Fax, 'Not Available') AS 'Fax Info'
FROM Supplier

SELECT OBJECT_ID('Customer'), OBJECT_ID('Order')
SELECT OBJECT_NAME(565577053)

SELECT SUSER_ID()
SELECT SUSER_NAME(259)
SELECT SUSER_SID()
SELECT SUSER_SNAME()

SELECT USER_ID()
SELECT USER_NAME(1)
```

Results	Messages
(No column name)	
1	5
(No column name)	
1	Northwind
(No column name)	
1	6916
(No column name)	
1	DMSPRO-ANHNT

Results	Messages
(No column name)	
1	259
(No column name)	
1	DMSpro-AnhNT\TuanTA
(No column name)	
1	0x0105000000000005150000004BB734FEB9E435E9240B1...
(No column name)	
1	DMSpro-AnhNT\TuanTA
(No column name)	
1	1
(No column name)	
1	dbo

Results	Messages
(No column name)	(No column name)
1	565577053 597577167
(No column name)	
1	Customer

100 %

1 Customer

Results Messages

	Id	CompanyName	ContactName	ContactTitle	City	Country	Phone	Fax	Fax Info
1	1	Exotic Liquids	Charlotte Cooper	NULL	London	UK	(171) 555-2222	NULL	Not Available
2	2	New Orleans Cajun Delights	Shelley Burke	NULL	New Orleans	USA	(100) 555-4822	NULL	Not Available
3	3	Grandma Kelly's Homestead	Regina Murphy	NULL	Ann Arbor	USA	(313) 555-5735	(313) 555-3349	(313) 555-3349
4	4	Tokyo Traders	Yoshi Nagase	NULL	Tokyo	Japan	(03) 3555-5011	NULL	Not Available

Hàm hệ thống



- Xem thông tin của các bảng trong một Database: ***SELECT * FROM sys.tables***

Results Messages

	name	object_id	principal_id	schema_id	parent_object_id	type	type_desc	create_date	modify_date
1	Customer	565577053	NULL	1	0	U	USER_TABLE	2017-03-06 10:09:38.830	2017-03-06 10:09:38.887
2	Order	597577167	NULL	1	0	U	USER_TABLE	2017-03-06 10:09:38.863	2017-03-06 10:09:38.890
3	OrderItem	661577395	NULL	1	0	U	USER_TABLE	2017-03-06 10:09:38.870	2017-03-06 10:09:38.893
4	Product	725577623	NULL	1	0	U	USER_TABLE	2017-03-06 10:09:38.877	2017-03-06 10:09:38.893
5	Supplier	789577851	NULL	1	0	U	USER_TABLE	2017-03-06 10:09:38.883	2017-03-06 10:09:38.893
6	OrderByMonth	1077578877	NULL	1	0	U	USER_TABLE	2017-03-22 16:38:05.047	2017-03-22 16:38:05.047

- Xem thông tin thao tác trên một bảng

```
SELECT [TableName] = OBJECT_NAME(object_id),  
last_user_update, last_user_seek, last_user_scan, last_user_lookup  
FROM sys.dm_db_index_usage_stats  
WHERE database_id = DB_ID('Northwind')  
AND OBJECT_NAME(object_id) = 'Product'
```

Results		Messages			
	TableName	last_user_update	last_user_seek	last_user_scan	last_user_lookup
1	Product	NULL	NULL	2017-03-22 16:55:24.263	2017-03-16 09:11:31.077
2	Product	NULL	2017-03-16 09:11:31.077	2017-03-16 09:11:31.077	NULL
3	Product	NULL	NULL	2017-03-22 16:38:05.013	NULL

Hàm hệ thống



- Chúng ta có thể kiểm tra xem view '***sys.dm_db_index_usage_stats***' của hệ thống bằng lệnh ***sp_helptext***

```
sp_helptext 'sys.dm_db_index_usage_stats'  
GO
```

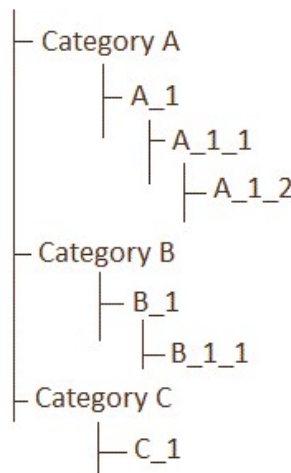
100 %


	Text
1	CREATE VIEW sys.dm_db_index_usage_stats AS
2	SELECT database_id, object_id, index_id,
3	user_seeks, user_scans, user_lookups, user_updates,
4	last_user_seek, last_user_scan, last_user_lookup, last_user_update,
5	system_seeks, system_scans, system_lookups, system_updates,
6	last_system_seek, last_system_scan, last_system_lookup, last_system_update
7	FROM OpenRowSet(TABLE LOGINDEXSTATS)
8	
9	-- Hiding Hekaton tables
10	WHERE status = 0 and ISNULL(OBJECTPROPERTY(object_id, 'TablesMemoryOptimized'), 0) = 0

Đệ quy với WITH



- Lệnh WITH cho phép ta thực hiện một vòng lặp đệ quy. Việc sử dụng nó trên dữ liệu dạng cây sẽ cho thấy tính tiện dụng của lệnh này. Cấu trúc dữ liệu dạng cây (cha-con) không xa lạ với những người sử dụng máy tính



Column Name	Data Type	Allow Nulls
 id	bigint	<input type="checkbox"/>
name	nvarchar(512)	<input type="checkbox"/>
description	nvarchar(1024)	<input checked="" type="checkbox"/>
parent_id	bigint	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Như trong hình, ta có nhiều category khác nhau, mỗi cái trong số đó lại có một hoặc nhiều con. Vậy mỗi category, ta sẽ có trường ID và Name, mô tả cho category đó. Còn việc tạo ra mối quan hệ cha con, ta phải thêm vào trường Parent_id, trường này sẽ trỏ tới ID của category mà nó phụ thuộc.

Đệ quy với WITH



- Trước tiên, ta sẽ lấy ra các phần tử Root, là những phần tử trên cùng của cây, có **Parent_Id** bằng **Null**, và sẽ có level bằng 0.

```
Select id, name, 0 as aLevel  
From Category  
Where parent_id is null
```

- Với các phần tử có level bằng 1 thì phải có Parent_Id bằng với ID của phần tử level 0. Có nghĩa là, **phần tử level thứ n, sẽ có Parent_Id bằng với ID của phần tử level thứ n-1.**
- Lợi dụng điều này, ta sẽ gọi đệ quy dùng WITH để tính level như sau:

Đệ quy với WITH



```
WITH temp(id, name, alevel)
  as (
    Select id, name, 0 as aLevel
    From Category
    Where parent_id is null

    Union All

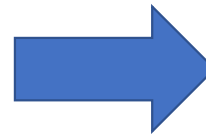
    Select b.id, b.name, a.alevel + 1
    From temp as a, Category as b
    Where a.id = b.parent_id
  )
Select * From temp
```

Đệ quy với WITH



- Khi đó, ta sẽ thu được danh sách category với các level tương ứng.

id	name	description	parent_id
13	Category A	NULL	NULL
14	Category B	NULL	NULL
15	Category C	NULL	NULL
16	A_1	NULL	13
17	A_1_1	NULL	16
18	A_1_2	NULL	17
19	B_1	NULL	14
20	B_1_1	NULL	19
21	C_1	NULL	15
NULL	NULL	NULL	NULL



	id	name	alevel
1	13	Category A	0
2	14	Category B	0
3	15	Category C	0
4	21	C_1	1
5	19	B_1	1
6	20	B_1_1	2
7	16	A_1	1
8	17	A_1_1	2
9	18	A_1_2	3

Truy vấn với WITH (CTE)



- - Sử dụng để định nghĩa khối các câu truy vấn phức tạp được sử dụng nhiều lần trong các câu lệnh truy vấn và cập nhật dữ liệu

```
WITH Luong_tung_phong AS (  
    SELECT ph.tenphong, SUM (nv.mucluong) AS Tong_phong  
    FROM nhanvien nv, phong ph  
    WHERE nv.maphong = ph.maphong  
    GROUP BY ph.tenphong),  
    TB_luong_cac_phong AS (  
    SELECT SUM (Tong_phong)/COUNT(*) AS TB_phong  
    FROM Luong_tung_phong)  
SELECT *  
FROM Luong_tung_phong  
WHERE Tong_phong > (SELECT TB_phong FROM TB_luong_cac_phong)  
ORDER BY tenphong
```

CTE (Common Table Expression)



- - CTE có thể được xem như một bảng chứa dữ liệu tạm thời từ câu lệnh được định nghĩa trong phạm vi của nó. CTE tương tự như một bảng dẫn xuất (derived table) ở chỗ nó không được lưu trữ như một đối tượng và chỉ kéo dài trong suốt thời gian của câu truy vấn.
- - Không giống như bảng dẫn xuất, CTE có thể tự tham chiếu tới bản thân của nó và có thể tham chiếu nhiều lần trong một câu truy vấn
- - **Mục đích của CTE**
 - Tạo truy vấn đệ quy (recursive query).
 - Thay thế View trong một số trường hợp.
 - Cho phép nhóm một cột từ truy vấn con.
 - Tham chiếu tới bảng kết quả nhiều lần trong cùng một lệnh.

CTE (Common Table Expression)



- **Ưu điểm của CTE**

- - CTE có nhiều ưu điểm như khả năng đọc dữ liệu được cải thiện và dễ dàng bảo trì các truy vấn phức tạp. Các truy vấn có thể được phân thành các khối nhỏ, đơn giản. Những khối này được sử dụng để xây dựng các CTE phức tạp hơn cho đến khi tập hợp kết quả cuối cùng được tạo ra.
 - CTE có thể được định nghĩa trong function, store procedure, view, trigger.

- **Cú pháp của CTE**

- WITH expression_name [(column_name [,...n])]
- AS (CTE_query_definition)
- SELECT
- FROM expression_name;

CTE (Common Table Expression)



```
WITH Sales_CTE (SalesPersonID, SalesOrderID, SalesYear)
AS
-- Define the CTE query.
(
    SELECT SalesPersonID, SalesOrderID, YEAR(OrderDate) AS SalesYear
    FROM Sales.SalesOrderHeader
    WHERE SalesPersonID IS NOT NULL
)
-- Define the outer query referencing the CTE name.
SELECT SalesPersonID, COUNT(SalesOrderID) AS TotalSales, SalesYear
FROM Sales_CTE
GROUP BY SalesYear, SalesPersonID
ORDER BY SalesPersonID, SalesYear;
GO
```

Ghép dòng bằng FOR XML PATH()

- Ví dụ bạn có bảng:

ProductID	CustomerName
1	Tuấn
1	Minh
1	Linh
2	Ngọc
2	Hiền

- Bạn muốn kết quả ra như sau:

ProductID	CustomerName
1	Tuấn, Minh, Linh
2	Ngọc, Hiền



Ghép dòng bằng FOR XML PATH()



- Bạn có thể dùng câu lệnh này:

```
SELECT DISTINCT C2.ProductID,  
    SUBSTRING(  
        (  
            SELECT ', '+C1.CustomerName AS [TEXT()]  
            FROM dbo.Customer C1  
            WHERE C1.ProductID = C2.ProductID  
            ORDER BY C1.ProductID  
            FOR XML PATH ('')  
        ), 2, 1000) CustomerList  
FROM dbo.Customer C2
```

ProductID	CustomerName
1	Tuấn, Minh, Linh
2	Ngọc, Hiền

Ghép dòng bằng FOR XML PATH()



- Ví dụ tiếp theo giả sử ta có dữ liệu như sau :

```
SELECT      CAT.Name AS [Category],  
            SUB.Name AS [Sub Category]  
FROM        Production.ProductCategory CAT  
INNER JOIN  Production.ProductSubcategory SUB  
ON          CAT.ProductCategoryID = SUB.ProductCategoryID
```

	Category	Sub Category
1	Bikes	Mountain Bikes
2	Bikes	Road Bikes
3	Bikes	Touring Bikes
4	Components	Handlebars
5	Components	Bottom Brackets
6	Components	Brakes
7	Components	Chains
8	Components	Cranksets

Ghép dòng bằng FOR XML PATH()



- Ghép các dòng theo cột Sub Category ta dùng

```
SELECT      CAT.Name AS [Category],
            STUFF((      SELECT ', ' + SUB.Name AS [text()]
                        -- Add a comma (,) before each value
                        FROM Production.ProductSubcategory SUB
                        WHERE
                        SUB.ProductCategoryID = CAT.ProductCategoryID
                        FOR XML PATH('') -- Select it as XML
                        ), 1, 1, '' )
            -- This is done to remove the first character
            -- from the result
            AS [Sub Categories]
FROM      Production.ProductCategory CAT
```

	Category	Sub Categories
1	Accessories	Bike Racks,Bike Stands,Bottles and Cages,Cleaners...
2	Bikes	Mountain Bikes,Road Bikes,Touring Bikes
3	Clothing	Bib-Shorts,Caps,Gloves,Jerseys,Shorts,Socks,Tights...
4	Components	Handlebars,Bottom Brackets,Brakes,Chains,Crankse...

Lưu trữ dữ liệu dạng XML



- - Ngoài cách lưu trữ dữ liệu trong các hệ quản trị CSDL ra thì bạn có thể lưu trữ dữ liệu trong file TXT, file JSON hay file XML đều được. Tuy nhiên với những hệ thống lớn thì bắt buộc ta phải lưu trữ trong hệ quản trị CSDL bởi vì nó cũng cấp những tính năng giúp quản lý dữ liệu tốt hơn.
- - Còn đối với XML hay JSON thì ứng dụng lớn nhất của nó trong lập trình web đó là xây dựng các Service và API, nghĩa là các API đó sẽ trả kết quả về dạng JSON hoặc XML các hệ thống khác có thể hiểu được.

Ví dụ để tạo một ứng dụng đặt phòng trên mobile thì bạn phải xây dựng một Service và nhiệm vụ của service đó là trả kết quả danh sách phòng về cho App Mobile, mà với ngôn ngữ lập trình Mobile khác hoàn toàn với PHP hay C# nên ta phải trao đổi dữ liệu thông qua XML hoặc JSON

Lưu trữ dữ liệu dạng XML



- Giả sử chúng ta có bảng dữ liệu Customer sau:

	Column Name	Data Type	Allow Nulls
🔑	CustomerId	int	<input type="checkbox"/>
	FirstName	varchar(50)	<input checked="" type="checkbox"/>
	LastName	varchar(50)	<input checked="" type="checkbox"/>
	Address	varchar(200)	<input checked="" type="checkbox"/>
	Email	varchar(200)	<input checked="" type="checkbox"/>
	PhoneNumber	varchar(50)	<input checked="" type="checkbox"/>

	CustomerId	FirstName	LastName	Address	Email	PhoneNumber
	1	Priyanka	Srivastava	lucknow	priyanka@yaho...	23232232
	2	Manish	Sharma	mumbai	manish@gmail.com	232323
	3	Kanak	Srivastava	Delhi	kanak@hotmail....	2165273
	4	Rachita	Pandey	Bhopal	Rachita@abc.com	7868768
	5	Jitender	Singh	Patiala	Jitender@abc.com	897879

Lưu trữ dữ liệu dạng XML



- Với cách viết cơ bản như sau:
- ***select * from Customer for xml auto***
- Ta có kết quả xml sau

```
<dbo.Customer CustomerId="1" FirstName="Priyanka" LastName="Srivastava" Address="lucknow" Email="priyanka@yahoo.com" PhoneNumber="23232232" />

<dbo.Customer CustomerId="2" FirstName="Manish" LastName="Sharma" Address="mumbai" Email="manish@gmail.com" PhoneNumber="232323" />

<dbo.Customer CustomerId="3" FirstName="Kanak" LastName="Srivastava" Address="Delhi" Email="kanak@hotmail.com" PhoneNumber="2165273" />

<dbo.Customer CustomerId="4" FirstName="Rachita" LastName="Pandey" Address="Bhopal" Email="Rachita@abc.com" PhoneNumber="7868768" />

<dbo.Customer CustomerId="5" FirstName="Jitender" LastName="Singh" Address="Patiala" Email="Jitender@abc.com" PhoneNumber="897879" />
```


Lưu trữ dữ liệu dạng XML



- Ta có thể chọn lọc cột để xuất XML. @ cho ta định nghĩa thuộc tính

```
SELECT  CustomerId AS '@Id',  
        FirstName AS '@FirstName',  
        LastName AS '@LastName'  
FROM dbo.Customer FOR XML PATH('Customer')
```

Resultant XML will be:

```
<Customer Id="1" FirstName="Priyanka" LastName="Srivastava" />  
<Customer Id="2" FirstName="Manish " LastName="Sharma" />  
<Customer Id="3" FirstName="Kanak" LastName="Srivastava" />  
<Customer Id="4" FirstName="Rachita" LastName="Pandey" />  
<Customer Id="5" FirstName="Jitender" LastName="Singh" />
```

Lưu trữ dữ liệu dạng XML



- Kết các cột lại thành một thuộc tính

```
SELECT CustomerId AS '@Id',  
FirstName + ' ' + LastName AS '@Name'  
FROM dbo.Customer FOR XML PATH('Customer')
```

Resultant XML will be :

```
<Customer Id="1" Name="Priyanka Srivastava" />  
<Customer Id="2" Name="Manish Sharma" />  
<Customer Id="3" Name="Kanak Srivastava" />  
<Customer Id="4" Name="Rachita Pandey" />  
<Customer Id="5" Name="Jitender Singh" />
```

Lưu trữ dữ liệu dạng XML



- Tạo Tag cha (root)

```
SELECT CustomerId AS '@Id',  
FirstName + ' ' + LastName AS '@Name'  
FROM dbo.Customer FOR XML PATH('Customer'), ROOT ('MyCustomers')
```

Resultant XML will be :

```
<MyCustomers>  
  <Customer Id="1" Name="Priyanka Srivastava" />  
  <Customer Id="2" Name="Manish Sharma" />  
  <Customer Id="3" Name="Kanak Srivastava" />  
  <Customer Id="4" Name="Rachita Pandey" />  
  <Customer Id="5" Name="Jitender Singh" />  
</MyCustomers>
```

Lưu trữ dữ liệu dạng XML



- Tạo Tag con

```
SELECT  CustomerId AS '@Id',
        FirstName + ' ' + LastName AS '@Name',
        [Address] AS 'ContactDetails/@PostalAddress',
        Email AS 'ContactDetails/@EmailAddress',
        PhoneNumber AS 'ContactDetails/@PhoneNumber'
FROM dbo.Customer FOR XML PATH('Customer'), ROOT ('MyCustomers')
```

Resultant XML will be:

```
<MyCustomers>
  <Customer Id="1" Name="Priyanka Srivastava">
    <ContactDetails PostalAddress="lucknow" EmailAddress="priyanka@yahoo.com"
    PhoneNumber="23232232" />
  </Customer>
  <Customer Id="2" Name="Manish Sharma">
    <ContactDetails PostalAddress="mumbai" EmailAddress="manish@gmail.com" Ph
oneNumber="232323" />
  </Customer>
```

Lưu trữ dữ liệu dạng XML

- Tạo Tag con



```
SELECT  CustomerId AS '@Id',  
        FirstName + ' ' + LastName AS '@Name',  
        [Address] AS 'ContactDetails/PostalAddress',  
        Email AS 'ContactDetails/EmailAddress',  
        PhoneNumber AS 'ContactDetails/PhoneNumber'  
FROM dbo.Customer FOR XML PATH('Customer'), ROOT ('MyCustomers')
```

```
<MyCustomers>  
  <Customer Id="1" Name="Priyanka Srivastava">  
    <ContactDetails>  
      <PostalAddress>lucknow</PostalAddress>  
      <EmailAddress>priyanka@yahoo.com</EmailAddress>  
      <PhoneNumber>23232232</PhoneNumber>  
    </ContactDetails>  
  </Customer>
```

Try ... Catch



- Đầu tiên, ta xét một ví dụ phát sinh lỗi đơn giản sau:

```
BEGIN TRY
    -- Generate some error.
    Declare @str varchar(20);
    Set @str = 'SQL SERVER!';
    print convert(datetime, @str);
END TRY
BEGIN CATCH
    SELECT
        ERROR_NUMBER() AS ErrorNumber
        ,ERROR_SEVERITY() AS ErrorSeverity
        ,ERROR_STATE() AS ErrorState
        ,ERROR_PROCEDURE() AS ErrorProcedure
        ,ERROR_LINE() AS ErrorLine
        ,ERROR_MESSAGE() AS ErrorMessage;
END CATCH;
GO
```

Tạo một câu lệnh SQL phát sinh ra lỗi

Đọc thông tin về lỗi

- Sau khi thực thi câu lệnh, ta thu được kết quả:

	ErrorNumber	ErrorSeverity	ErrorState	ErrorProcedure	ErrorLine	ErrorMessage
1	241	16	1	NULL	5	Conversion failed when converting dateti...

Try ... Catch



- Dưới đây là một số hàm cung cấp thông tin về lỗi vừa phát sinh:

STT	TÊN HÀM	CHỨC NĂNG
1	ERROR_NUMBER()	Trả lại mã lỗi (dưới dạng số)
2	ERROR_SEVERITY()	Trả lại mức độ nghiêm trọng của lỗi
3	ERROR_STATE()	Trả lại trạng thái của lỗi (dưới dạng số)
4	ERROR_PROCEDURE()	Trả lại tên của Stored Procedure hoặc tên của Trigger đã phát sinh lỗi
5	ERROR_LINE()	Trả lại vị trí dòng lệnh đã phát sinh ra lỗi.
6	ERROR_MESSAGE()	Trả lại thông báo lỗi dưới hình thức văn bản (text)

Try ... Catch



- Ví dụ:

```
-- Khai báo biến:
DECLARE @v_a float = 20;
DECLARE @v_b float = 0;
DECLARE @v_c float;
DECLARE @v_Error_Number integer;

-- Sử dụng BEGIN TRY .. END TRY để bắt lỗi.
-- Nếu lỗi xảy ra trong khối này
-- nó sẽ nhảy vào khối BEGIN CATCH .. END CATCH.
BEGIN TRY

    ---
    PRINT '@v_a = ' + CAST(@v_a AS varchar(15));
    PRINT '@v_b = ' + CAST(@v_b AS varchar(15));
    --Lỗi chia cho 0 xảy ra tại đây.
    SET @v_c = @v_a / @v_b;

    -- Dòng bên dưới này sẽ không được chạy.
    -- Chương trình nhảy vào khối BEGIN CATCH .. END CATCH
    PRINT '@v_c= ' + CAST(@v_c AS varchar(15));

END TRY

-- BEGIN CATCH .. END CATCH phải được đặt ngay
-- phía sau của khối BEGIN TRY .. END TRY.
```


Try ... Catch



```
-- BEGIN CATCH .. END CATCH phải được đặt ngay
-- phía sau của khối BEGIN TRY .. END TRY.
BEGIN CATCH
    -- Mã lỗi.
    SET @v_Error_Number = ERROR_NUMBER();
    -- In ra mã lỗi:
    PRINT 'Error Number: ' + CAST(@v_Error_Number AS varchar(15));
    -- Nguyên nhân lỗi:
    PRINT 'Error Message: ' + ERROR_MESSAGE();
    -- Mức độ nghiêm trọng của lỗi:
    PRINT 'Error Severity: ' + CAST(ERROR_SEVERITY() AS varchar(15));
    -- Mã trạng thái:
    PRINT 'Error State: ' + CAST(ERROR_STATE() AS varchar(15));
    -- Dòng bị lỗi:
    PRINT 'Error Line: ' + CAST(ERROR_LINE() AS varchar(15));
    -- Tên của thủ tục (hoặc function) hoặc trigger, có code gây ra lỗi này.
    PRINT 'Error Procedure: ' + ERROR_PROCEDURE();
END CATCH;

END;
```

```
100 % <
Messages
@v_a = 20
@v_b = 0
Error Number: 8134
Error Message: Divide by zero error encountered.
Error Severity: 16
Error State: 1
Error Line: 62
```



THANK YOU

Dr. Tran Anh Tuan, Faculty of Mathematics and Computer Science, University of
Science, HCMC