Database Management System
Advanced SQL Commands P1

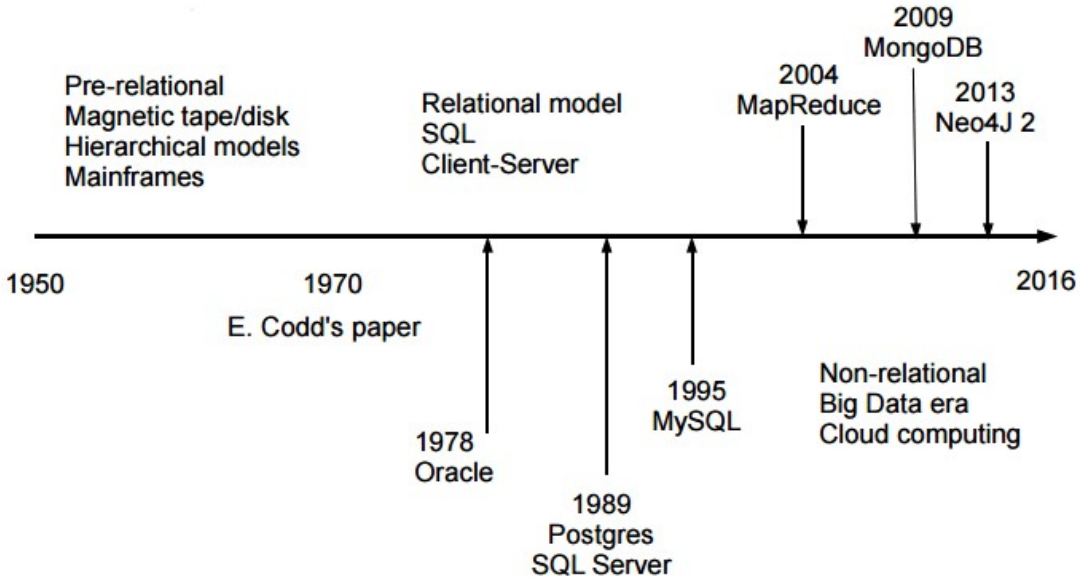
Dr. Tran Anh Tuan,

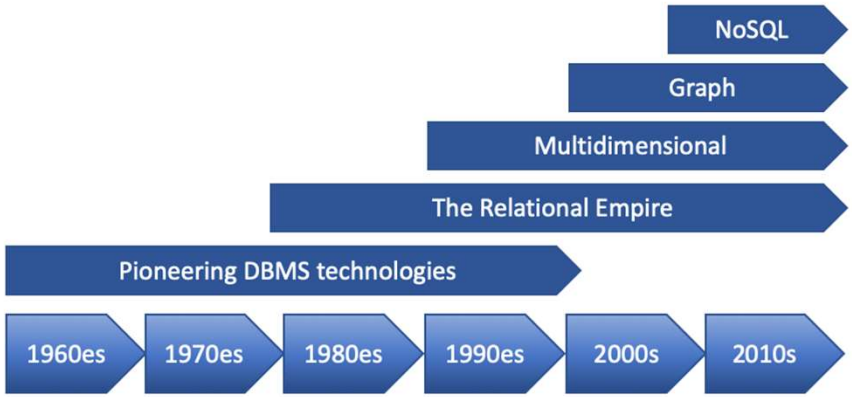
Faculty of Mathematics and Computer Science,

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.



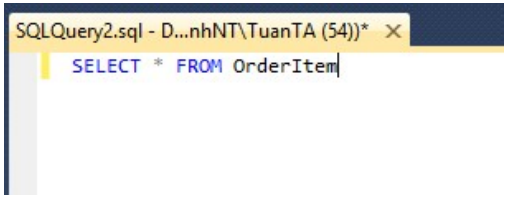


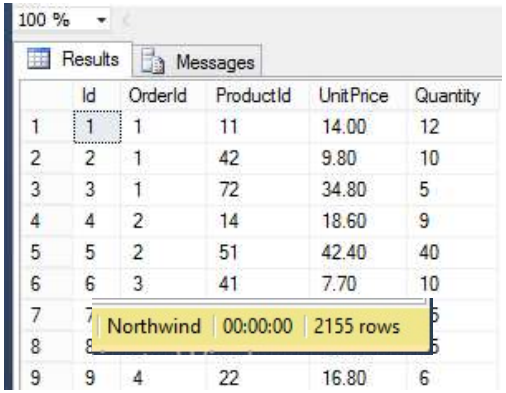
Syllabus

- 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

Phân trang trong SQL

- - Câu hỏi đặt ra ở đây, là mình muốn lấy dữ liệu từ vị trí dòng 50 -100 hoặc chỉ lấy ra 1 record ở vị trí 2017 thì mình phải làm sao ? Nếu như các bạn nào từng làm bên lập trình web, thì chúng ta sẽ sử dụng câu truy vấn này để phân trang SQL SERVER





Phân trang trong SQL

```
SELECT *
FROM (
   SELECT *, ROW_NUMBER() OVER (ORDER BY Id) AS RowNum
   FROM OrderItem
) AS MyDerivedTable
WHERE MyDerivedTable.RowNum BETWEEN 50 AND 100
```

100 %

Results | Messages

	Id	OrderId	ProductId	UnitPrice	Quantity	RowNum
1	50	18	17	31.20	30	50
2	51	18	70	12.00	20	51
3	52	19	12	30.40	12	52
4	53	20	40	14.70	50	53
5	54	20	59	44.00	70	54
6	55	20	76	14.40	15	55
7	56	21	29	99.00	10	56
8	57	21	72	27.80	4	57
9	58	22	33	2.00	60	58
10	59	22	72	27.80	20	59
11	60	23	36	15.20	30	60

Query executed successfully.

Phân trang trong SQL

```
SELECT *
FROM (
   SELECT *, ROW_NUMBER() OVER (ORDER BY Id) AS RowNum
   FROM OrderItem
) AS MyDerivedTable
WHERE MyDerivedTable.RowNum BETWEEN 2000 AND
MyDerivedTable.RowNum
```

Nếu bạn nào muốn select từ vị trí 2000 đến cuối dòng thì mình viết sql như bên dưới

100 %

Results | Messages

	Id	OrderId	ProductId	UnitPrice	Quantity	RowNum
146	2145	830	39	18.00	2	2145
147	2146	830	41	9.65	3	2146
148	2147	830	46	12.00	3	2147
149	2148	830	52	7.00	2	2148
150	2149	830	55	24.00	2	2149
151	2150	830	60	34.00	2	2150
152	2151	830	64	33.25	2	2151
153	2152	830	66	17.00	1	2152
154	2153	830	73	15.00	2	2153
155	2154	830	75	7.75	4	2154
156	2155	830	77	13.00	2	2155

Query executed successfully.

Phân trang trong SQL

- Xuất danh sách các khách hàng gồm No, Id, FullName, City, Country, Phone và sắp xếp theo FullName

```
SELECT *
FROM (
SELECT ROW_NUMBER() OVER (ORDER BY FirstName + ' ' + LastName) AS [No], Id,
FirstName + ' ' + LastName AS FullName, City, Country, Phone
                FROM Customer
) AS ReportTable
```

100 %

Results | Messages

	No	Id	FullName	City	Country	Phone
1	1	69	Alejandra Camino	Madrid	Spain	(91) 745 6200
2	2	52	Alexander Feuer	Leipzig	Germany	0342-023176
3	3	2	Ana Trujillo	México D.F.	Mexico	(5) 555-4729
4	4	81	Anabela Domingues	Sao Paulo	Brazil	(11) 555-2167
5	5	31	André Fonseca	Campinas	Brazil	(11) 555-9482
6	6	19	Ann Devon	London	UK	(171) 555-0297
7	7	41	Annette Roulet	Toulouse	France	61.77.61.10
8	8	3	Antonio Moreno	México D.F.	Mexico	(5) 555-3932
9	9	21	Aria Cruz	Sao Paulo	Brazil	(11) 555-9857
10	10	75	Art Braunschweiger	Lander	USA	(307) 555-4680
11	11	61	Bernardo Batista	Rio de Janeiro	Brazil	(21) 555-4252

Phân trang trong SQL

- PARTITION BY giúp đánh dấu dòng theo tuần tự từng nhóm
- Xuất danh sách hóa đơn của mỗi khách hàng theo Total Amount có kèm số dòng

```
SELECT
 ROW_NUMBER() OVER (PARTITION BY CustomerId ORDER BY TotalAmount
DESC) AS [RowNumber],  CustomerId, OrderNumber,
OrderDate,TotalAmount
FROM [Order]
```

100 %

Results | Messages

	RowNumber	CustomerId	OrderNumber	OrderDate	TotalAmount
1	1	1	542773	2013-08-25 00:00:00.000	1086.00
2	2	1	543141	2014-04-09 00:00:00.000	960.00
3	3	1	542822	2013-10-03 00:00:00.000	878.00
4	4	1	542965	2014-01-15 00:00:00.000	851.00
5	5	1	543082	2014-03-16 00:00:00.000	491.20
6	6	1	542832	2013-10-13 00:00:00.000	330.00
7	1	2	543056	2014-03-04 00:00:00.000	514.40
8	2	2	542755	2013-08-08 00:00:00.000	479.75
9	3	2	542889	2013-11-28 00:00:00.000	320.00
10	4	2	542438	2012-09-18 00:00:00.000	88.80
11	1	3	542665	2013-05-13 00:00:00.000	2156.50

Phân trang trong SQL

- Xuất danh sách top 3 hóa đơn của mỗi khách hàng theo Total Amount có kèm số dòng

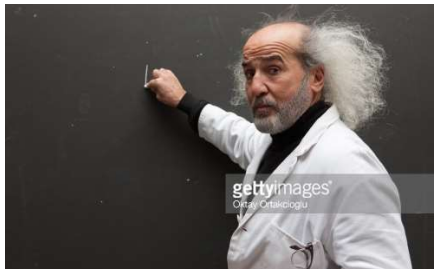
```
SELECT *
FROM (
SELECT ROW_NUMBER() OVER (PARTITION BY CustomerId ORDER BY
TotalAmount DESC) AS [RowNumber], CustomerId, OrderNumber,
OrderDate,TotalAmount
                FROM [Order]
) AS ReportTable
WHERE ReportTable.RowNumber <= 3
```

100 %

Results | Messages

	RowNumber	CustomerId	OrderNumber	OrderDate	TotalAmount
1	1	1	542773	2013-08-25 00:00:00.000	1086.00
2	2	1	543141	2014-04-09 00:00:00.000	960.00
3	3	1	542822	2013-10-03 00:00:00.000	878.00
4	1	2	543056	2014-03-04 00:00:00.000	514.40
5	2	2	542755	2013-08-08 00:00:00.000	479.75
6	3	2	542889	2013-11-28 00:00:00.000	320.00
7	1	3	542665	2013-05-13 00:00:00.000	2156.50
8	2	3	542703	2013-06-19 00:00:00.000	2082.00
9	3	3	542807	2013-09-22 00:00:00.000	956.90
10	1	4	543083	2014-03-16 00:00:00.000	4675.00
11	2	4	542688	2013-06-04 00:00:00.000	2142.90

PIVOT TABLE



- - *PIVOT*, trong các hệ quản trị cơ sở dữ liệu, là lệnh tổng hợp dữ liệu cho phép chuyển dữ liệu trong một cột của một Table thành các trường dữ liệu của một Table khác.
- - Lệnh này cho phép người dùng có thể chọn một trường dữ liệu làm tiêu chí, từ đó "chiếu" các dữ liệu khác lên trường dữ liệu này để quan sát.

SELECT * FROM Sales_Order;
Kết quả lệnh Select có dạng như sau:

	Customer	Product	Quantity
1	Minh Thu	Laptop	3
2	Minh Thu	iPhone	2
3	Minh Thu	Laptop	5
4	Tuan Anh	Laptop	3
5	Tuan Anh	iPhone	3
6	Tuan Anh	iPhone	4

PIVOT TABLE

- - Sau khi đã có đầy đủ dữ liệu rồi, ta thực hiện việc thống kê thông tin về tổng số hàng mà mỗi khách hàng đặt mua cho mỗi loại hàng. Sử dụng lệnh PIVOT áp dụng cho cột Product và tính tổng cột Quantity, ta được:

```
SELECT * FROM Sales_Order
PIVOT (SUM(Quantity) FOR Product IN ([Laptop],[iPhone])) AS PivotedOrder;
```

Thực thi câu lệnh SQL này, ta được:

	Customer	Laptop	iPhone
1	Minh Thu	8	2
2	Tuan Anh	3	7

- - Lưu ý, ta có thể Select tiếp từ kết quả của PIVOT trên như sau:

```
SELECT Customer, [Laptop] FROM
(SELECT * FROM Sales_Order) AS OriginalOrder
PIVOT (SUM(Quantity) FOR Product IN ([Laptop],[iPhone])) AS PivotedOrder;
```

	Customer	Laptop
1	Minh Thu	8
2	Tuan Anh	3

PIVOT TABLE

- - Xuất dữ liệu CustomerId kèm theo tổng số lượng các hóa đơn theo tháng của khách hàng đó

```
SELECT * FROM [Order]
```

100 %

Results | Messages

	Id	OrderDate	OrderNumber	CustomerId	TotalAmount
1	1	2012-07-04 00:00:00.000	542378	85	440.00
2	2	2012-07-05 00:00:00.000	542379	79	1863.40
3	3	2012-07-08 00:00:00.000	542380	34	1813.00
4	4	2012-07-08 00:00:00.000	542381	84	670.80
5	5	2012-07-09 00:00:00.000	542382	76	3730.00
6	6	2012-07-10 00:00:00.000	542383	34	1444.80
7	7	2012-07-11 00:00:00.000	542384	14	625.20
8	8	2012-07-12 00:00:00.000	542385	68	2490.50

PIVOT TABLE

- - Ta cần tính tổng hóa đơn theo tháng của mỗi Customer trước và đưa vào một bảng tạm tên là OrderByMonth. Tuy nhiên phải chú ý là bảng OrderByMonth phải tồn tại.
- - Dùng câu lệnh sau để kiểm tra xem bảng có tồn tại trong DB hay chưa

```
SELECT * FROM INFORMATION_SCHEMA.TABLES
      WHERE TABLE_NAME = N'OrderByMonth'
```

100 %

Results | Messages

	TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	TABLE_TYPE
1	Northwind	dbo	OrderByMonth	BASE TABLE

Hay dùng OBJECT_ID

```
IF OBJECT_ID(N'dbo.OrderByMonth', N'U') IS NOT NULL
BEGIN
  ......--Do something
END
```

PIVOT TABLE

- Nếu tồn tại rồi thì ta xóa dữa liệu bảng đó trước khi đưa dữ liệu mới vào

```
IF EXISTS(SELECT * FROM INFORMATION_SCHEMA.TABLES
      WHERE TABLE_NAME = N'OrderByMonth')
BEGIN
      TRUNCATE TABLE OrderByMonth
END

SELECT CustomerId, MONTH(OrderDate) as MonthOrder, COUNT(OrderNumber) AS
OrderSum
INTO OrderByMonth
FROM [Order] GROUP BY CustomerId, MONTH(OrderDate)
```

100 %

Results | Messages

	CustomerId	MonthOrder	OrderSum
1	1	1	1
2	3	1	1
3	5	1	2
4	6	1	1
5	7	1	1
6	9	1	1
7	10	1	3
8	12	1	1
9	16	1	1
10	17	1	1

PIVOT TABLE

- Dùng PIVOT Table để xuất danh sách các khách hàng và số hóa đơn theo tháng

```
SELECT      CustomerByMonth.CustomerId,
            ISNULL(CustomerByMonth.[1],0) AS [Thang 1],
            ISNULL(CustomerByMonth.[2],0) AS [Thang 2],
            ISNULL(CustomerByMonth.[3],0) AS [Thang 3],
            ISNULL(CustomerByMonth.[4],0) AS [Thang 4],
            ISNULL(CustomerByMonth.[5],0) AS [Thang 5],
            ISNULL(CustomerByMonth.[6],0) AS [Thang 6],
            ISNULL(CustomerByMonth.[7],0) AS [Thang 7],
            ISNULL(CustomerByMonth.[8],0) AS [Thang 8],
            ISNULL(CustomerByMonth.[9],0) AS [Thang 9],
            ISNULL(CustomerByMonth.[10],0) AS [Thang 10],
            ISNULL(CustomerByMonth.[11],0) AS [Thang 11],
            ISNULL(CustomerByMonth.[12],0) AS [Thang 12]
FROM
(
        SELECT * FROM OrderByMonth
        PIVOT (SUM(OrderSum) FOR MonthOrder IN
([1],[2],[3],[4],[5],[6],[7],[8],[9],[10],[11],[12])) AS PivotedOrder
) CustomerByMonth
```

100 %

Results | Messages

	CustomerId	Thang 1	Thang 2	Thang 3	Thang 4	Thang 5	Thang
1	1	1	0	1	1	0	0
2	2	0	0	1	0	0	0
3	3	1	0	0	1	1	1
4	4	0	2	2	1	0	1
5	5	2	4	1	0	1	1
6	6	1	0	1	3	0	1
7	7	1	2	0	0	0	3
8	8	0	0	1	0	0	0
9	9	1	2	3	1	2	0
10	10	3	0	5	4	0	0
11	11	0	0	4	1	2	1
12	12	1	1	1	2	0	0
13	13	0	0	0	0	0	0
14	14	0	0	1	3	0	0
15	15	0	0	2	2	0	0
16	16	1	1	1	0	0	0
17	17	1	0	0	1	1	0

UNPIVOT

- Lệnh UNPIVOT có chức năng ngược lại với chức năng của lệnh PIVOT. Xét ví dụ sau:

```
select * from Sales_PivotedOrder;
```

Thực thi các lệnh trên, ta được bảng có dạng sau:

	Customer	Laptop	iPhone
1	Minh Thu	8	2
2	Tuan Anh	3	7

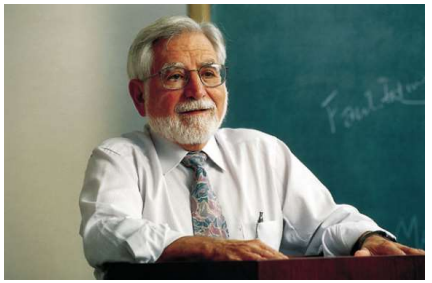
- Thực thi lệnh UNPIVOT trên bảng Sales_PivotedOrder trên:

```
SELECT Customer, Product, Quantity FROM Sales_PivotedOrder
UNPIVOT (Quantity FOR Product IN ([Laptop],[iPhone])) AS UnPVT;
```

Ta thu được kết quả:

	Customer	Product	Quantity
1	Minh Thu	Laptop	8
2	Minh Thu	iPhone	2
3	Tuan Anh	Laptop	3
4	Tuan Anh	iPhone	7

CASE WHEN



CASE trong SQL dùng để đánh giá một danh sách các điều kiện và trả về 1 trong các biểu thức kết quả thỏa mãn điều kiện đánh giá

- CASE có 2 định dạng:
- - Chức năng CASE đơn giản hay còn gọi là simple CASE
- - Chức năng CASE tìm kiếm hay còn gọi là searched CASE
- ***Trong đó:***
- - Simple CASE là so sánh một biểu thức với một bộ các biểu thức đơn giản để xác định kết quả.
- - Searched CASE là đánh giá một bộ các biểu thức Boolean để xác định kết quả
- Cả 2 định dạng trên đều hỗ trợ đối số ELSE (nhưng không bắt buộc)

CASE WHEN

- Cú pháp

Simple CASE:

```
CASE biểu_thức_đầu_vào
   WHEN biểu_thức_1 THEN biểu_thức_kết_quả_1
   WHEN biểu_thức_2 THEN biểu_thức_kết_quả_2
   …
   WHEN biểu_thức_n THEN biểu_thức_kết_quả_3
   ELSE biểu_thức_kết_quả
END
```

Searched CASE:

```
CASE
   WHEN biểu_thức_điều-kiện_1 THEN biểu_thức_kết_quả_1
   WHEN biểu_thức_ điều-kiện _2 THEN biểu_thức_kết_quả_2
   …
   WHEN biểu_thức_ điều-kiện _n THEN biểu_thức_kết_quả_n
   ELSE biểu_thức_kết_quả
END
```

CASE WHEN

- Xuất danh sách các hóa đơn kèm theo thông tin OrderMonth

```
SELECT  Id, OrderNumber, OrderDate,
    (CASE MONTH(OrderDate)
        WHEN 1 THEN 'Thang 1'
        WHEN 2 THEN 'Thang 2'
        WHEN 3 THEN 'Thang 3'
        WHEN 4 THEN 'Thang 4'
        WHEN 5 THEN 'Thang 5'
        WHEN 6 THEN 'Thang 6'
        WHEN 7 THEN 'Thang 7'
        WHEN 8 THEN 'Thang 8'
        WHEN 9 THEN 'Thang 9'
        WHEN 10 THEN 'Thang 10'
        WHEN 11 THEN 'Thang 11'
        ELSE 'Thang 1'
     END
    ) AS OrderMonth
FROM [Order]
```

100 %

Results | Messages

	Id	OrderNumber	OrderDate	OrderMonth
1	1	542378	2012-07-04 00:00:00.000	Thang 7
2	2	542379	2012-07-05 00:00:00.000	Thang 7
3	3	542380	2012-07-08 00:00:00.000	Thang 7
4	4	542381	2012-07-08 00:00:00.000	Thang 7
5	5	542382	2012-07-09 00:00:00.000	Thang 7
6	6	542383	2012-07-10 00:00:00.000	Thang 7
7	7	542384	2012-07-11 00:00:00.000	Thang 7
8	8	542385	2012-07-12 00:00:00.000	Thang 7
9	9	542386	2012-07-15 00:00:00.000	Thang 7
10	10	542387	2012-07-16 00:00:00.000	Thang 7

Query executed successfully.

CASE WHEN

- Ví dụ: Searched CASE

```
SELECT  Id, OrderNumber, OrderDate,
     (CASE
         WHEN MONTH(OrderDate) < 4 THEN 'Quy 1'
         WHEN MONTH(OrderDate) < 7 THEN 'Quy 2'
         WHEN MONTH(OrderDate) < 10 THEN 'Quy 3'
         ELSE 'Quy 4'
      END
     ) AS OrderPeriod
FROM [Order]
```

100 %

Results | Messages

	Id	OrderNumber	OrderDate	OrderMonth
1	1	542378	2012-07-04 00:00:00.000	Quy 3
2	2	542379	2012-07-05 00:00:00.000	Quy 3
3	3	542380	2012-07-08 00:00:00.000	Quy 3
4	4	542381	2012-07-08 00:00:00.000	Quy 3
5	5	542382	2012-07-09 00:00:00.000	Quy 3
6	6	542383	2012-07-10 00:00:00.000	Quy 3
7	7	542384	2012-07-11 00:00:00.000	Quy 3
8	8	542385	2012-07-12 00:00:00.000	Quy 3
9	9	542386	2012-07-15 00:00:00.000	Quy 3
10	10	542387	2012-07-16 00:00:00.000	Quy 3

Query executed successfully.

Các hàm chuyển đổi kiểu dữ liệu trong SQL

- Hàm CAST
- - Với cú pháp hàm CAST bên dưới cho phép bạn có thể chuyển đổi một biểu thức nào đó sang một kiểu dữ liệu bất kỳ mong muốn.

Ví dụ: Để hiển thị danh sách các vật tư có trong bảng VATTU, trong đó có cột tỷ lệ phần trăm được hiển thị theo dạng xxx%. Bạn sử dụng hàm CAST để chuyển đổi giá trị cột phần trăm từ kiểu dữ liệu số sang kiểu dữ liệu chuỗi và sử dụng toán tử cộng chuỗi (+) để nối thêm ký tự %

```
SELECT MAVTU, TENVTU
       TYLE = CAST(PHANTRAM AS VARCHAR(3))+"%"
FROM VATTU
```

MAVTU	TENVTU	TYLE
BI01	Bia lon Tiger	60%
DD01	Đầu DVD Hitachi 1 đĩa	40%
DD02	Đầu DVD Hitachi 3 đĩa	40%
LO01	Loa Panasonic 1000W	10%

Các hàm chuyển đổi kiểu dữ liệu trong SQL

- Hàm CONVERT
- - Với cú pháp hàm CONVERT bên dưới cho phép bạn có thể chuyển đổi một biểu thức nào đó sang một kiểu dữ liệu bất kỳ mong muốn nhưng có thể theo một định dạng nào đó (đặc biệt đối với kiểu dữ liệu ngày).
- Cú pháp : `CONVERT (Kiểu_dữ_liệu, Biểu_thức [,Định_danh])`
- • **Kiểu dữ liệu :** tên kiểu dữ liệu mà biểu thức sẽ được chuyển đổi sang.
- • **Biểu thức :** là tên của cột bên trong bảng hoặc một biểu thức tính toán muốn chuyển sang kiểu dữ liệu mới.
- • **Định dạng :** là một con số chỉ định việc định dạng cho việc chuyển đổi dữ liệu từ dạng ngày sang dạng chuỗi. Bảng bên dưới mô tả một số định dạng thường dùng trong hàm CONVERT.

Các hàm chuyển đổi kiểu dữ liệu trong SQL

- Để hiển thị chi tiết và đếm tổng số các đơn đặt hàng theo từng năm tháng. Bạn sử dụng hàm CONVERT để chuyển đổi giá trị cột ngày đặt hàng từ kiểu dữ liệu ngày sang chuỗi.

```
SELECT NAMTHANG=CONVERT(CHAR(6), NGAYDH, 112),
       SODH, CONVERT(CHAR(10), NGAYDH, 103) AS NGAYDH
FROM DONDH
ORDER BY NAMTHANG
COMPUTE COUNT(SODH) BY NAMTHANG
```

```
NAMTHANG  SODH      NGAYDH
--------------------------------------
200201    D001      15/01/2002
200201    D002      30/01/2002
```

Định dạng năm (yy)	Định dạng năm (yyyy)	Hiển thị dữ liệu
1	101	mm/dd/yy
2	102	yy-mm-dd
3	103	dd/mm/yy
4	104	dd.mm.yy
5	105	dd-mm-yy
6	106	dd mon yy
7	107	mon dd,yy
8	108	hh:mm:ss
9	109	mon dd yyyy hh:mm:ss
10	110	mm-dd-yy
11	111	yy/mm/dd
12	112	Yymmdd
13	113	dd mon yyyy hh:mm:ss
14	114	hh:mm:ss:mmm
	21 hoặc 121	yyyy-mm-dd hh:mi:ss.mmm
	20 hoặc 120	yyyy-mm-dd hh:mi:ss

Các hàm chuyển đổi kiểu dữ liệu trong SQL

- Hàm STR
- - Với cú pháp hàm STR bên dưới cho phép bạn có thể chuyển đổi kiểu dữ liệu số sang kiểu dữ liệu chuỗi. Phải đảm bảo đủ vùng trắng để chứa các ký số khi chuyển đổi sang kiểu dữ liệu chuỗi
- Cú pháp : STR (Số_thực, Số_ký_tự [,Số_lẻ]) → Chuỗi
- • **Số thực :** là một biểu thức có kiểu dữ liệu số thực.
- • **Số ký tự :** số vùng trắng dùng để chứa các ký số sau khi chuyển sang kiểu dữ liệu chuỗi.
- • **Số lẻ :** chỉ định số thập phân.
- • **Chuỗi :** là chuỗi ký tự kết quả chứa các ký số sau khi đã chuyển đổi kiểu dữ liệu số thành kiểu dữ liệu chuỗi.

Các hàm chuyển đổi kiểu dữ liệu trong SQL

- - Để liệt kê danh sách các vật tư đã nhập trong tháng 01/2002 có hiển thị thêm cột đơn vị tính. Bạn sử dụng hàm STR để chuyển đổi giá trị cột số lượng nhập từ kiểu dữ liệu số sang kiểu dữ liệu chuỗi và kết hợp toán tử cộng chuỗi (+) để nối thêm giá trị dữ liệu cột đơn vị tính.

```
SELECT PN.SOPN, SLNHAP = STR(SLNHAP, 5) + " " + DVTINH,
        TENVTU
      FROM CTPNHAP CTPN
      INNER JOIN VATTU VT
      ON VT.MAVTU=CTPN.MAVTU
      INNER JOIN PNHAP PN
      ON PN.SOPN=CTPN.SOPN
      WHERE CONVERT(CHAR(6), NGAYNHAP, 112)="200201"
```

```
SOPN SLNHAP   TENVTU
-----------------------------------------------
N001  8 Bộ      Đầu DVD Hitachi 1 đĩa
N001  10 Bộ     Đầu DVD Hitachi 3 đĩa
N002  2 Bộ      Đầu DVD Hitachi 1 đĩa
N002  5 Bộ      Đầu DVD Hitachi 3 đĩa

(5 row(s) affected)
```

Một số hàm xử lý chuỗi trong SQ

- – **ASCII**
  Hàm này trả về giá trị ASCII của keyboard ví dụ (@,R,9,*)
  Cú pháp
  – ASCII (character)
  Ví dụ:
  SELECT ASCII(‘a’) — giá trị trả về= 97
  SELECT ASCII(‘b’) — giá trị trả về= 98

  – **SPACE**
  Hàm này trả về khoảng trống trong câu lệnh sql
  Cú pháp
  – SPACE (integer)
  Ví dụ:
  SELECT (‘SQL’) + SPACE(0) + (‘TUTORIALS’)– giá trị trả về= SQLTUTORIALS
  SELECT (‘SQL’) + SPACE(1) + (‘TUTORIALS’)– giá trị trả về = SQL TUTORIALS

Một số hàm xử lý chuỗi trong SQL

- – **CHARINDEX**
  Trả về vị trí được tìm thấy của một chuỗi trong chuỗi cha
  Cú pháp
  – CHARINDEX (string1, string2 [, start_location])
  Ví dụ:
  SELECT CHARINDEX('SQL', 'Well organized understand SQL tutorial')– Value = 27
  SELECT CHARINDEX('SQL', 'Well organized understand SQL tutorial', 30)– Value = 0 (bởi vì giá trị bắt đầu truyền vào từ ký tự 30 trở đi)

- – **REPLACE**
  Hàm thay thế chuỗi
  Ví dụ:
  Cú pháp
  – REPLACE ('string1' , 'string2' , 'string3')
  SELECT REPLACE('All Function' , 'All', 'SQL')– Value = SQL Function

Một số hàm xử lý chuỗi trong SQ

- – **UPPER, LOWER**
  Hàm chuyển đổi thành chữ hoa và chữ thường
  Ví dụ:
  Cú pháp – UPPER(‘string1’) – LOWER(‘string1’)
  SELECT UPPER(‘Khong con mua thu’)– Value = ‘KHONG CON MUA THU’
  SELECT LOWER(‘kHONG CON MUA THU’)– Value = ‘khong con mua thu’
- – **LEFT,RIGHT,SUBSTRING**
  Hàm cắt chuỗi bên trái, phải và ở giữa
  Ví dụ:
  Cú pháp
  – LEFT(‘string1’, số kí tự) – RIGHT(‘string1’, số kí tự)
  – SUBSTRING (‘string1’, vị trí, số kí tự)
  SELECT LEFT(‘Khong con mua thu’,5)– Value = ‘Khong’
  SELECT RIGHT(‘kHONG CON MUA THU’,3)– Value = ‘THU’
  SELECT SUBSTRING (‘kHONG CON MUA THU’,6,3)– Value = ‘CON’

Một số hàm xử lý chuỗi trong SC

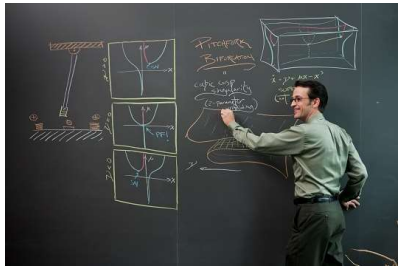
- – **LTRIM, RTRIM**
  Loại bỏ khoảng trắng bên trái, bên phải
  Ví dụ:
  Cú pháp – LTRIM(‘string1’) – RTRIM (‘string1’)
  SELECT LTRIM(‘ Khong con mua thu’)– Value = ‘Khong con mua thu’
  SELECT RTRIM (‘kHONG CON MUA THU ‘)– Value = ‘kHONG CON MUA THU’
- – **LEN**
  Trả về số ký tự trong chuỗi
  Ví dụ:
  Cú pháp
  – LEN(‘string’)
  SELECT LEN(‘Khong con mua thu’)– Value = 17

Một số hàm xử lý chuỗi trong SQ

- – **REVERSE** Đảo chuối
  Ví dụ:
  Cú pháp
  – REVERSE(‘string’)
  SELECT REVERSE(‘Khong con mua thu’)– Value = ‘uht aum noc gnohK’
- – **STUFF**
  Với cú pháp hàm STUFF bên dưới có kết quả trả về là một chuỗi mới sau khi đã hủy bỏ một số ký tự hiện có và thêm vào một chuỗi con khác tại vị trí vừa hủy bỏ
  Cú pháp
  – STUFF (‘string’,vị trí, chiều dài, chuỗi con)
  vị trí: là một số nguyên chỉ định vị trí bắt đầu hủy bỏ các ký tự bên trong chuỗi nguồn.
  chiều dài: là một số nguyên chỉ định bao nhiêu ký tự sẽ bị hủy bỏ trong chuỗi nguồn đếm từ bên trái vị trí chỉ định.

  SELECT STUFF (‘123456789’,4,3,’ABDCEF’)– Value = ‘123ABCDEF789’

Các hàm Ranking



- **Các hàm Ranking là gì?**

  - Các hàm Ranking cho phép bạn có thể đánh số liên tục (xếp loại) cho các tập hợp kết quả. Các hàm này có thể được sử dụng để cung cấp số thứ tự trong hệ thống đánh số tuần tự khác nhau. Có thể hiểu đơn giản như sau: bạn có từng con số nằm trên từng dòng liên tục, tại dòng thứ nhất xếp loại số 1, dòng thứ 2 xếp loại số là 2… Bạn có thể sử dụng hàm ranking theo các nhóm số tuần tự, mỗi một nhóm sẽ được đánh số theo lược đồ 1,2,3 và nhóm tiếp theo lại bắt đầu bằng 1,2,3…

- Hàm ROW_NUMBER

Hàm đầu tiên nói tới là ROW_NUMBER. Hàm này trả lại một dãy số tuần tự bắt đầu từ 1 cho mỗi dòng hay nhóm trong tập hợp kết quả. Hàm ROW_

```
ROW_NUMBER ( ) OVER ( [ <Mệnh đề PARTITION BY> ] <Mệnh đề ORDER BY>)
```

Các hàm Ranking

- Để hiểu thêm về cách sử dụng hàm ROW_NUMBER, ví dụ dưới sẽ đánh số liên tục cho tất cả các dòng trong bảng Person và sắp xếp chúng theo trường Age
-

```
SELECT ROW_NUMBER() OVER (ORDER BY Age) AS [Row Number by Age],
      FirstName,
      Age
 FROM Person
```

Và đây là tập hợp kết quả mã T-SQL trên:

Row Number by Age	FirstName	Age
1	Larry	5
2	Doris	6
3	George	6
4	Mary	11
5	Sherry	11
6	Sam	17
7	Ted	23
8	Marty	23
9	Sue	29
10	Frank	38
11	John	40

Các hàm Ranking

- - Giả sử bạn không muốn tập hợp kết quả của bạn được sắp xếp mà muốn đưa bảng trở lại sắp xếp theo số bản ghi của từng dòng. Hàm ROW_NUMBER lại luôn yêu cầu phải có mệnh đề ORDER BY, vậy bạn cần phải đưa một giá trị nào đó vào trong mệnh đề này.

```
SELECT ROW_NUMBER() OVER (ORDER BY (SELECT 1)) AS [Row Number by
Record Set],
      FirstName,
      Age
  FROM Person
```

Đây là tập hợp kết quả khi chạy hàm truy vấn trên:

Row Number by Record	FirstName	Age
1	Ted	23
2	John	40
3	George	6
4	Mary	11
5	Sam	17
6	Doris	6
7	Frank	38
8	Larry	5
9	Sue	29
10	Sherry	11
11	Marty	23

Các hàm Ranking

- - Hàm ROW_NUMBER không chỉ cho phép bạn sắp xếp toàn bộ tập hợp dòng mà còn có thể sử dụng mệnh đề PARTITION để lọc ra nhóm dòng cần đánh số. Các dòng sẽ được đánh số tuần tự trong từng giá trị PARTITION độc nhất. Các dãy số được đánh sẽ luôn bắt đầu từ 1 cho từng giá trị PARTITION mới trong tập hợp bản ghi của bạn.

```
SELECT ROW_NUMBER() OVER (PARTITION BY Gender ORDER BY Age) AS
[Partition by Gender],  FirstName,  Age, Gender
  FROM Person
```

Khi chạy truy vấn trên, tập hợp kết quả sẽ ra như sau:

Partition by Gender	FirstName	Age	Gender
1	Doris	6	F
2	Mary	11	F
3	Sherry	11	F
4	Sue	29	F
1	Larry	5	M
2	George	6	M
3	Sam	17	M
4	Ted	23	M

Các hàm Ranking

- **Hàm RANK**
  Đôi khi bạn muốn một dòng có cùng sắp xếp giá trị cột như các dòng khác có cùng một xếp loại. Nếu thế thì hàm RANK () có thể giúp bạn. Hàm RANK có cú pháp như sau:
- RANK ( ) OVER ([<Mệnh đề PARTITION BY>] <Mệnh đề ORDER BY>)
- ***Trong đó***:

  <Mệnh đề PARTITION BY> là một cột hay tập hợp các cột được sử dụng để quyết đinh việc đánh số liên tục trong hàm RANK

  <Mệnh đề ORDER BY> là một cột hay tập hợp các cột được sử dụng để sắp xếp tập hợp kết quả trong nhóm (partition)

Các hàm Ranking

- - Hàm RANK sẽ đánh số liên tục một tập hợp bản ghi nhưng khi có 2 dòng có cùng giá trị sắp xếp thì hàm sẽ đánh giá là cùng bậc giá trị. Giá trị xếp loại vẫn sẽ tăng kể cả khi có 2 dòng cùng giá trị, vì vậy khi đánh giá một giá trị sắp xếp tiếp theo thì số thứ tự vẫn tiếp tục được đánh nhưng sẽ tăng thêm 1 giá trị vào các dòng tiếp theo trong tập hợp.

```
SELECT RANK() OVER (ORDER BY Age) AS [Rank by Age],
      FirstName,
      Age
 FROM Person
```

Và kết quả trả về:

Rank by Age	FirstName	Age
1	Larry	5
2	Doris	6
2	George	6
4	Mary	11
4	Sherry	11
6	Sam	17
7	Ted	23
7	Marty	23
9	Sue	29
10	Frank	38
11	John	40

Các hàm Ranking

- - Nếu bạn muốn có một nhiều xếp loại trong tập hợp bản ghi của mình thì với từng xếp loại bạn cần đặt một nhóm cụ thể bằng cách sử dụng mệnh đề PARTITION BY trong hàm RANK. Ví dụ dưới sẽ cho thấy tác dụng khi tôi nhóm xếp loại theo Gender và sắp xếp theo Age

```
SELECT RANK() OVER (PARTITION BY Gender ORDER BY Age) AS [Partition by Gender],
      FirstName,
      Age,
      Gender
  FROM Person
```

Đây là kết quả khi chạy các hàm truy vấn trên:

Partition by Gender	FirstName	Age	Gender
1	Doris	6	F
2	Mary	11	F
2	Sherry	11	F
4	Sue	29	F
1	Larry	5	M
2	George	6	M
3	Sam	17	M
4	Ted	23	M
4	Marty	23	M
6	Frank	38	M
7	John	40	M

Các hàm Ranking

- **Hàm DENSE_RANK**
- - Hàm DENSE_RANK cũng giống như hàm RANK, tuy vậy, hàm này không cung cấp khoảng cách giữa các số xếp loại. Thay vào đó, hàm này sẽ xếp loại liên tục cho từng giá trị ORDER BY cụ thể. Với hàm DENSE_RANK, kể cả khi có hai dòng có cùng giá trị xếp loại thì dòng tiếp theo vẫn chỉ tăng thêm một giá trị so với dòng trên. Hàm DENSE_RANK có cú pháp như hàm RANK

```
SELECT DENSE_RANK() OVER (ORDER BY Age) AS [Dense Rank by Age],
    FirstName,
    Age
 FROM Person
```

Đoạn mã trên sẽ xuất ra như sau:

```
Dense Rank by Age  FirstName Age
------------------- ---------- -----------
1         Larry   5
2         Doris   6
2         George  6
3         Mary    11
3         Sherry  11
4         Sam     17
5         Ted     23
5         Marty   23
6         Sue     29
7         Frank   38
8         John    40
```

Các hàm Ranking

- **Hàm NTILE**
- - Hàm cuối cùng là hàm NTILE. Đây là hàm được sử dụng để phá vỡ tập hợp bản ghi trong một số cụ thể của các nhóm. Hàm NTILE cũng sử dụng cú pháp như các hàm ranking khác.

```
SELECT FirstName,
      Age,
      NTILE(3) OVER (ORDER BY Age) AS [Age Groups]
  FROM Person
```

```
FirstName Age    Age Groups
---------- ----------- --------------------
Larry       5      1
Doris       6      1
George      6      1
Mary        11     1
Sherry      11     2
Sam         17     2
Ted         23     2
Marty       23     2
Sue         29     3
Frank       38     3
John        40     3
```

Các hàm Ranking

- Hàm NTILE là một hàm rất có ích nếu bạn chỉ muốn trả lại một nhóm cụ thể trong các bản ghi. Dưới đây là một ví dụ khi tôi muốn trả lại chỉ nhóm người có độ tuổi chung bình (Nhóm Age 2) từ ví dụ trên.

```
SELECT FirstName,
      Age,
      Age AS [Age Group]
FROM ( SELECT FirstName,
            Age,
            NTILE(3) OVER (ORDER BY Age) AS AgeGroup
       FROM Person) A
WHERE AgeGroup = 2
```

Kết quả của câu lệnh trên:

FirstName	Age	Age Group
Sherry	11	11
Sam	17	17
Ted	23	23
Marty	23	23

THANK YOU