Quick Link

1- Giới thiệu

2- Công cụ làm việc với SQL Server

3- Cơ sở dữ liệu học SQL (LearningSQL)

4- Các câu lệnh truy vấn (Query)

4.1- SQL Select

4.2- SQL Distinct

4.3- SQL Where

4.4- SQL And Or (Và, hoặc)

4.5- SQL IN (Trong phạm vi..)

4.6- SQL Between (Nằm giữa ...)

4.7- SQL Wildcard

4.8- SQL Like (Giống với ...)

4.9- SQL Order By (Sắp xếp bởi)

4.10- SQL Group By (Nhóm bởi ...)

4.11- SQL Having

5- Các câu lệnh trèn dữ liệu (Insert)

5.1- Câu lệnh Insert Into

5.2- Câu lệnh Insert Into Select

6- Câu lệnh cập nhập (Update)

7- Câu lệnh xóa dữ liệu (Delete)

8- SQL Functions

8.1- SQL Count

8.2- SQL Sum

8.3- SQL AVG

8.4- SQL MIN

8.5- SQL MAX

9- SQL Join

9.1- INNER JOIN (Hoặc JOIN)

9.2- LEFT OUTER JOIN (Hoặc LEFT JOIN)

9.3- RIGHT OUTER JOIN (Hoặc RIGHT JOIN)

9.4- FULL OUTER JOIN (Hoặc OUTER JOIN)

10- Câu truy vấn con (Subquery)

10.1- Subquery trong mệnh đề Where

10.2- Subquery trong mệnh đề From

10.3- Subquery trong mệnh đề Select

11- Lập trình SQL Server Transact-SQL

1- Giới thiệu

Tài liệu này hướng dẫn cho những người mới bắt đầu làm quen với SQL. Tài liệu hướng dẫn dựa trên:

SQL Server (2008 - 2014)

Nội dung bao gồm

Ngôn ngữ SQL

Ngữ pháp SQL (Ngữ pháp cơ bản, tiêu chuẩn cho mọi loại DB)

Ngữ pháp SQL riêng của SQLServer.

Xem thêm:

[Hướng dẫn học SQL cho người mới bắt đầu với MySQL](http://o7planning.org/vi/10237/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-mysql)

[Hướng dẫn học SQL cho người mới bắt đầu với Oracle](http://o7planning.org/vi/10231/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-oracle)

2- Công cụ làm việc với SQL Server

Ở đây tôi sử dụng SQLServer Management Studio, một công cụ sẵn có sau khi bạn cài đặt SQLServer. Bạn có thể xem qua hướng dẫn cài đặt SQLServer tại:

[Hướng dẫn cài đặt và cấu hình SQL Server Express 2014](http://o7planning.org/vi/10299/huong-dan-cai-dat-va-cau-hinh-sql-server-express-2014)

[Hướng dẫn cài đặt và cấu hình SQL Server Express 2012](http://o7planning.org/vi/10297/huong-dan-cai-dat-va-cau-hinh-sql-server-express-2012)

[Hướng dẫn cài đặt và cấu hình SQL Server Express 2008](http://o7planning.org/vi/10219/huong-dan-cai-dat-va-cau-hinh-sql-server-express-2008)

3- Cơ sở dữ liệu học SQL (LearningSQL)

Tài liệu này sử dụng cơ sở dữ liệu LearningSQL (phiên bản SQLServer). Bạn cần tạo cơ sở dữ liệu đó để tiện cho việc học (Bạn chỉ mất không quá 5 phút để làm việc này).   
  
Script tạo cơ sở dữ liệu và cấu trúc của cơ sở dữ liệu này bạn có thể xem tại:

[Cơ sở dữ liệu SQL Server mẫu để học SQL](http://o7planning.org/vi/10241/co-so-du-lieu-sql-server-mau-de-hoc-sql)

4- Các câu lệnh truy vấn (Query)

4.1- SQL Select

Câu lệnh Select là một câu lệnh cơ bản nhất trong SQL, mục đích là truy vấn dữ liệu trong các bảng (Table).

PRODUCT\_TYPE: Bảng dữ liệu các kiểu sản phẩm (Service type of bank).

Đây là câu lệnh truy vấn dữ liệu trên bảng PRODUCT\_TYPE

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | -- Truy vấn dữ liệu trên bảng Product\_Type  -- Truy vấn toàn bộ các cột.    Select \* From Product\_Type;    -- Câu lệnh truy vấn trên và câu lệnh dưới đây là tương đương nhau.  -- Pty là alias (tên bí danh đặt cho bảng Product\_Type).    Select Pty.\* From Product\_Type Pty;    -- Truy vấn một vài cột  -- Có thể liệt kê ra.    Select Pty.Product\_Type\_Cd       ,Pty.Name  From   Product\_Type Pty; |

Kết quả truy vấn:

Truy vấn dữ liệu trên bảng EMPLOYEE:

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | -- Truy vấn 4 cột trên bảng Employee (Bảng nhân viên).  -- Sử dụng Emp là Alias (Tên bí danh) cho bảng Employee.    Select Emp.Emp\_Id       ,Emp.First\_Name       ,Emp.Last\_Name       ,Emp.Dept\_Id  From   Employee Emp; |

Kết quả truy vấn:

Đặt Alias cho cột:

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | -- Hàm Convert(Varchar, <intvalue>) chuyển một số thành chữ (Varchar) (Hàm của SQLServer)  -- Sử dụng toán tử + để nối hai chuỗi (Có thể không được hỗ trợ bởi DB khác)  -- Chúng ta có một cột mới sử dụng 'as' để định nghĩa tên cột cho cột mới này.  Select Emp.Emp\_Id      ,Emp.First\_Name      ,Emp.Last\_Name      ,Emp.Dept\_Id      ,'EMP' +  Convert(Varchar,Emp.Emp\_Id) As Emp\_No2  -- Cột mới  From   Employee Emp;    -- Có thể không cần sử dụng 'as' trong định nghĩa Alias cho cột.  Select Emp.Emp\_Id      ,Emp.First\_Name      ,Emp.Last\_Name      ,Emp.Dept\_Id      ,'EMP' + Convert(varchar,Emp.Emp\_Id ) Emp\_No2  -- Cột mới  From   Employee Emp; |

Kết quả truy vấn:

4.2- SQL Distinct

Câu lệnh distinct được sử dụng cùng với Select, dùng để select các dữ liệu, bỏ qua các dữ liệu trùng nhau. Cú pháp là:

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1 | Select distinct <column1>, <column2>  .... |

Xem ví dụ:

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | -- Truy vấn sản phẩm (Sản phẩm dịch vụ của Ngân hàng)  -- Cột mã sản phẩm, tên và kiểu sản phẩm.  Select Pro.Product\_Cd      ,Pro.Name      ,Pro.Product\_Type\_Cd  From   Product Pro;    -- Truy vấn các kiểu sản phẩm  (Product\_Type\_Cd) trong bảng Product.  -- Dữ liệu là nhiều, nhưng trùng nhau.  Select Pro.Product\_Type\_Cd from Product Pro;    -- Cần sử dụng Distinct để loại bỏ việc trùng lặp.  Select Distinct Pro.Product\_Type\_Cd from Product Pro; |

Kết quả chạy các câu lệnh trên:

4.3- SQL Where

Where là câu lệnh nhằm giới hạn phạm vi tìm kiếm. Chẳng hạn bạn muốn tìm kiếm những sản phẩm dịch vụ có kiểu "Cho vay cá nhân và kinh doanh".

Product\_Type\_Cd = 'LOAN'.

Bạn cần truy vấn trong bảng PRODUCT, tại các vị trí có Product\_Type\_Cd= 'LOAN'.

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4 | -- Truy vấn bảng Product tìm các sản phẩm dịch vụ có kiểu:  -- "Cho vay cá nhân và kinh doanh".  -- Ứng với cột Product\_Type\_Cd = 'LOAN'.  Select \* From Product Pro Where Pro.Product\_Type\_Cd = 'LOAN'; |

Kết quả truy vấn:

Một ví dụ khác sử dụng where và nhiều điều kiện kèm theo.

4.4- SQL And Or (Và, hoặc)

And và Or là 2 lệnh điều kiện sử dụng trong where:

Chẳng hạn bạn muốn tìm danh sách các nhân viên có tên bắt đầu bởi chữ 'S' và thuộc phòng điều hành (Operations).

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | -- Câu lệnh tìm kiếm các Nhân viên có tên bắt đầu bởi chữ S.  Select Emp.Emp\_Id       ,Emp.First\_Name       ,Emp.Last\_Name       ,Emp.Dept\_Id  From   Employee Emp  Where  Emp.First\_Name Like 'S%';    -- Câu lệnh tìm kiếm các nhân viên có tên bắt đầu bởi chữ S  -- và thuộc phòng điều hành (Operations)  Dept\_Id  = 1.    Select Emp.Emp\_Id       ,Emp.First\_Name       ,Emp.Last\_Name       ,Emp.Dept\_Id  From   Employee Emp  Where  Emp.First\_Name Like 'S%'  And    Emp.Dept\_Id = 1; |

Kết quả chạy các câu truy vấn:

Ví dụ:

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | -- Tìm kiếm các nhân viên có tên bắt đầu bởi chữ S hoặc P.  -- và trong phòng điều hành (Operations). (Dept\_Id = 1)  Select Emp.Emp\_Id       ,Emp.First\_Name       ,Emp.Last\_Name       ,Emp.Dept\_Id  From   Employee Emp  Where  (Emp.First\_Name Like 'S%' Or Emp.First\_Name Like 'P%')  And    Emp.Dept\_Id = 1; |

Kết quả chạy câu lệnh:

4.5- SQL IN (Trong phạm vi..)

Câu lệnh in trong where sử dụng trong tình huống tìm kiếm giá trị thuộc một tập hợp nhất định.

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | -- Câu lệnh này tìm kiếm các nhân viên có tên  -- hoặc Susan hoặc Paula hoặc Helen.  Select Emp.Emp\_Id       ,Emp.First\_Name       ,Emp.Last\_Name       ,Emp.Dept\_Id  From   Employee Emp  Where  Emp.First\_Name In ('Susan'                          ,'Paula'                          ,'Helen'); |

Kết quả chạy câu lệnh:

4.6- SQL Between (Nằm giữa ...)

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | -- Tìm các nhân viên có Emp\_Id trong khoảng từ 5 đến 10.  Select Emp.Emp\_Id      ,Emp.First\_Name      ,Emp.Last\_Name      ,Emp.Start\_Date  From   Employee Emp  Where  (Emp.Emp\_Id Between 5 And 10);    -- Câu lệnh trên tương đương với:  Select Emp.Emp\_Id       ,Emp.First\_Name       ,Emp.Last\_Name       ,Emp.Start\_Date  From   Employee Emp  Where  Emp.Emp\_Id >= 5  And    Emp.Emp\_Id <= 10; |

Kết quả câu lệnh:

Câu lệnh BETWEEN cũng được sử dụng trong việc giới hạn thời gian. Chẳng hạn bạn tìm kiếm các nhân viên bắt đầu làm trong khoảng thời gian:

03-05-2002 ==> 09-08-2002  (dd-MM-yyyy)

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | -- Câu lệnh này tìm kiếm các nhân viên bắt đầu vào làm việc trong 1 khoảng thời gian  -- xác định trong mệnh đề where.  -- 03-05-2002 ==> 09-08-2002  (Theo dd-MM-yyyy)  Select Emp.Emp\_Id       ,Emp.First\_Name       ,Emp.Last\_Name       ,Emp.Start\_Date       , -- Hàm Convert(Varchar, <Date>, 105) chuyển Date thành Varchar dạng DD-MM-YYYY        -- Đây là hàm của SQLServer. không dùng cho DB khác.        Convert(Varchar, Emp.Start\_Date               ,105) Start\_Date\_Vn  From   Employee Emp  Where  -- Hàm Convert(Datetime, <Varchar>, 105) chuyển text dạng DD-MM-YYYY sang Datetime  -- (Đây là hàm của SQLServer, có thể ko có trên các DB khác)  ( --   Emp.Start\_Date Between Convert(Datetime, '03-05-2002'                                 ,105) And   Convert(Datetime,'09-08-2002'          ,105) --  ); |

Kết quả chạy câu lệnh:

4.7- SQL Wildcard

Có 2 ký tự đặc biệt trong SQL:

Ký tự %

Ký tự \_

Ý nghĩa:

% mô tả 0, 1 hoặc nhiều ký tự bất kỳ.

\_ mô tả chính xác một ký tự.

Hai ký tự này thường được sử dụng trong điều kiện LIKE.

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | -- Tìm kiếm các Khách hàng (Customer) có số FED\_ID theo định dạng:  -- Phần trước bất kỳ, rồi tới dấu - , rồi tới 2 ký tự, rồi tới dấu -, và cuối bất kỳ.  -- Sử dụng 2 dấu \_ để minh họa rằng đó là 2 ký tự.  -- (Mỗi dấu \_ là chính xác một ký tự).    Select Cus.Cust\_Id       ,Cus.Fed\_Id       ,Cus.Address  From   Customer Cus  where cus.fed\_id like '%-\_\_-%'; |

Kết quả chạy ví dụ:

4.8- SQL Like (Giống với ...)

Câu lệnh này bạn đã quen thuộc qua các ví dụ ở trên.

4.9- SQL Order By (Sắp xếp bởi)

Việc query dữ liệu cho một tập kết quả, mà có thể nó sắp xếp không như ý muốn, sử dụng Order by để sắp xếp kết quả trả về.

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | -- Cú pháp    SELECT "column\_name"  FROM "table\_name"  [WHERE "condition"]  ORDER BY "column\_name1" [ASC, DESC], "column\_name2" [ASC, DESC];    -- Ghi chú:  -- ASC: nghĩa là sắp xếp tăng dần (Mặc định)  -- DESC: Nghĩa là sắp xếp giảm dần. |

Ví dụ:

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | -- Sắp xếp ưu tiên Product\_Type\_Cd tăng dần  -- Sau đó mới tới Name (Cũng tăng dần)  Select Pro.Product\_Cd       ,Pro.Product\_Type\_Cd       ,Pro.Name  From   Product Pro  Order  By Pro.Product\_Type\_Cd Asc          ,Pro.Name            Asc;      -- Trong Order BY, ASC là mặc định.  -- Vì vậy có thể ko cần viết ASC.  Select Pro.Product\_Cd       ,Pro.Product\_Type\_Cd       ,Pro.Name  From   Product Pro  Order  By Pro.Product\_Type\_Cd          ,Pro.Name;    -- Sắp xếp ưu tiên Product\_Type\_Cd giảm dần  -- Sau đó mới tới Name (Tăng dần)  Select Pro.Product\_Cd       ,Pro.Product\_Type\_Cd       ,Pro.Name  From   Product Pro  Order  By Pro.Product\_Type\_Cd Desc          ,Pro.Name            Asc; |

Kết quả chạy ví dụ:

Order By bao giờ cũng đứng sau where.

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | -- Tìm các nhân viên có tên bắt đầu bởi S.  -- Sắp xếp giảm dần theo ngày bắt đầu vào làm việc.  Select Emp.Emp\_Id       ,Emp.First\_Name       ,Emp.Last\_Name       ,Emp.Start\_Date  From   Employee Emp  Where  Emp.First\_Name Like 'S%'  Order  By Emp.Start\_Date Desc; |

Kết quả chạy câu lệnh:

Hoặc:

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | -- Sử dụng thứ tự của cột vào trong câu Order by.  -- First\_Name là cột thứ 2 trong câu Select  -- Có thể sử dụng: Order by 2 thay cho Order by First\_Name.  Select Emp.Emp\_Id      ,Emp.First\_Name      ,Emp.Last\_Name      ,Emp.Start\_Date  From   Employee Emp  Order  By 2 Desc; |

4.10- SQL Group By (Nhóm bởi ...)

Trước hết chúng ta cần hiểu các hàm tổng hợp ( Aggregate Functions) là gì:

Sum: Hàm tính tổng

Avg: Hàm lấy trung bình

Count: Hàm tính số lần

Min: Hàm tìm giá trị nhỏ nhất

Max: Hàm tìm giá trị lớn nhất

Đó là một số hàm tổng hợp ( Aggregate) thông dụng. Chúng có thể tham gia vào câu lệnh nhóm ( Group by).

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | -- Truy vấn dữ liệu trong bảng Account.    Select Acc.Account\_Id       ,Acc.Product\_Cd       ,Acc.Avail\_Balance       ,Acc.Pending\_Balance  From   Account Acc; |

Câu hỏi đặt ra bạn muốn xem tổng số tiền có trong tài khoản, ứng với mỗi loại dịch vụ ( Product\_Cd) khác nhau. Điều đó có nghĩa là bạn cần nhóm trên các Product\_Cd.

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | Select Acc.Product\_Cd       ,Count(Acc.Product\_Cd) As Count\_Acc -- Số tài khoản       ,Sum(Acc.Avail\_Balance) As Sum\_Avail\_Balance -- Tổng số tiền trong tài khoản       ,Avg(Acc.Avail\_Balance) As Avg\_Avail\_Balance -- Số tiền trung bình  From   Account Acc  Group  By Acc.Product\_Cd; |

Kết quả:

Như vậy bạn có một cái nhìn đánh giá:

Có 4 tài khoản tham gia dịch vụ "Tiền gửi tiết kiệm" (SAV) với tổng số tiền là 1855.76 và trung bình mỗi tài khoản có 463.94.

...

4.11- SQL Having

Mệnh đề HAVING cho phép bạn chỉ định các điều kiện lọc kết quả nhóm nào sẽ xuất hiện trong kết quả cuối cùng.   
  
Mệnh đề WHERE đặt các điều kiện vào các cột đã chọn, trong khi mệnh đề HAVING đưa ra các điều kiện đối với các nhóm được tạo bởi mệnh đề GROUP BY.

Giả sử bạn muốn nhóm các loại hình dịch vụ ( Product\_Cd) trên bảng Account, và chỉ hiển thị ra các loại hình nào có số người tham gia > 3.

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | Select Acc.Product\_Cd       ,Count(Acc.Product\_Cd) As Count\_Acc -- Số tài khoản       ,Sum(Acc.Avail\_Balance) As Sum\_Avail\_Balance -- Tổng số tiền trong tài khoản       ,Avg(Acc.Avail\_Balance) As Avg\_Avail\_Balance -- Số tiền trung bình  From   Account Acc  Group  By Acc.Product\_Cd  Having Count(Acc.Product\_Cd) > 3; |

Kết quả chạy ví dụ:

Phân biệt Where & Having

Bạn cần phân biệt Where và Having trong cùng một câu lệnh.

Where là câu lệnh lọc bớt dữ liệu trước khi nhóm (Group)

Having là câu lệnh lọc bớt dữ liệu sau khi đã nhóm (Group)

Trường hợp bạn muốn có các thông tin tổng hợp của một chi nhánh ngân hàng (Bảng BRANCH). Bạn có thể sử dụng where để lọc bớt dữ liệu trước khi group.

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | Select Acc.Product\_Cd       ,Count(Acc.Product\_Cd) As Count\_Acc       ,Sum(Acc.Avail\_Balance) As Sum\_Avail\_Balance       ,Avg(Acc.Avail\_Balance) As Avg\_Avail\_Balance  From   Account Acc  -- Sử dụng Where lọc bớt dữ liệu trước khi group.  Where  Acc.Open\_Branch\_Id = 1  Group  By Acc.Product\_Cd  -- Sử dụng Having lọc bớt dữ liệu sau khi đã group  Having Count(Acc.Product\_Cd) > 1; |

Kết quả chạy ví dụ:

5- Các câu lệnh trèn dữ liệu (Insert)

5.1- Câu lệnh Insert Into

Cú pháp:

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4 | -- Cú pháp câu lệnh Insert Into:    INSERT INTO "table\_name" ("column1", "column2", ...)  VALUES ("value1", "value2", ...); |

Ví dụ bạn trèn thêm một giao dịch khách hàng vào bảng ACC\_TRANSACTION:

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | -- Trèn 1 dòng dữ liệu vào bảng Acc\_Trasaction  -- Cột Txn\_ID tự động được sinh ra.  -- (Txn\_ID không tham gia vào trong câu Insert)  -- Current\_Timestamp là hàm của SQLServer lấy ra giờ hệ thống (System Date)  -- Current\_Timestamp : Giờ hiện tại.  Insert Into Acc\_Transaction    (Amount    ,Funds\_Avail\_Date    ,Txn\_Date    ,Txn\_Type\_Cd    ,Account\_Id    ,Execution\_Branch\_Id    ,Teller\_Emp\_Id)  Values    (100 -- Amount    ,Current\_Timestamp -- Funds\_Avail\_Date    ,Current\_Timestamp -- Txn\_Date    ,'CDT' -- Txn\_Type\_Cd    ,2 -- Account\_Id    ,Null -- Execution\_Branch\_Id    ,Null -- Teller\_Emp\_Id     ); |

5.2- Câu lệnh Insert Into Select

Bạn có thể sử dụng câu Select để cung cấp dữ liệu trèn vào bảng. Thông qua câu lệnh Insert Into ... Select.

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4  5 | -- Ngữ pháp câu lệnh INSERT INTO .. SELECT    INSERT INTO "table1" ("column1", "column2", ...)  SELECT "column3", "column4", ...  FROM "table2"; |

Ví dụ:

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | -- Txn\_Id tự sinh ra (Không cần phải tham gia vào câu Insert)  -- Trèn nhiều dòng dữ liệu vào bảng Acc\_Transaction  -- Lấy dữ liệu từ câu Select.    Insert Into Acc\_Transaction   ( Txn\_Date   ,Account\_Id   ,Txn\_Type\_Cd   ,Amount   ,Funds\_Avail\_Date)   Select Acc.Open\_Date -- Txn\_Date         ,Acc.Account\_Id -- Account\_Id         ,'CDT' -- Txn\_Type\_Cd         ,200 -- Amount         ,Acc.Open\_Date -- Funds\_Avail\_Date   From   Account Acc   Where  Acc.Product\_Cd = 'CD'; |

6- Câu lệnh cập nhập (Update)

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4  5 | -- Cú pháp câu lệnh Update:    UPDATE "table\_name"  SET "column\_1" = "new value 1", "column\_2"= "new value 2"  WHERE "condition"; |

Ví dụ bạn muốn tăng tiền trong các tài khoản của khách hàng có CUST\_ID = 1 lên 2%.

Câu lệnh Update:

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4  5 | -- Update, tăng số tiền trong tài khoản lên 2% cho khách hàng CUST\_ID = 1.  Update Account  Set    Avail\_Balance   = Avail\_Balance + 2 \* Avail\_Balance / 100       ,Pending\_Balance = Pending\_Balance + 2 \* Pending\_Balance / 100  Where  Cust\_Id = 1; |

Truy vấn lại, sau khi Update.

7- Câu lệnh xóa dữ liệu (Delete)

Cú pháp xóa dữ liệu trong bảng.

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4 | -- Cú pháp xóa các dòng dữ liệu trong bảng.    DELETE FROM "table\_name"  WHERE "condition"; |

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4  5 | -- Xóa 2 dòng dữ liệu trong bảng Acc\_Transaction:    Delete From Acc\_Transaction  Where Txn\_Id In (25                     ,26); |

8- SQL Functions

8.1- SQL Count

Count() là một hàm, đếm số dòng trong câu Query. Thông thường nó thường được sử dụng cùng với Group by.

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3 | -- Đếm số row (dòng) có trong bảng Account    Select Count(Acc.Account\_Id) As Count\_Acc From Account Acc; |

Đếm số tài khoản ngân hàng, có giao dịch với ngân hàng (Bảng Acc\_Transaction)

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3 | -- Đếm số tài khoản có giao dịch với ngân hàng.    Select Count(distinct txn.Account\_id) as Distinct\_Acc From Acc\_Transaction txn; |

Sử dụng với Group by:

Một khách hàng có thể mở nhiều tài khoản, mỗi tài khoản ứng với một sản phẩm (dịch vụ) của ngân hàng.

Bạn muốn liệt kê các khách hàng ( CUST\_ID) và tương ứng là số tài khoản họ đã mở.

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | -- Đếm số tài khoản đã mở ứng với mỗi khách hàng.    Select Acc.Cust\_Id       ,Count(Acc.Account\_Id) As Count\_Acc  From   Account Acc  Group  By Acc.Cust\_Id; |

8.2- SQL Sum

Sum() là hàm dùng để tính tổng giá trị một cột số.

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4 | -- Cú pháp:    SELECT SUM("column\_name")  FROM "table\_name"; |

Ví dụ:

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | -- Tính tổng số tiền trong các tài khoản của khách hàng với Cust\_ID = 1  Select Sum(Acc.Avail\_Balance)  As Sum\_Avail\_Balance  From Account Acc Where Acc.Cust\_Id = 1;        -- Sử dụng với Group by.  -- Tính tổng số tiền trong tài khoản ứng với mỗi khách hàng  Select Acc.Cust\_Id      ,Sum(Acc.Avail\_Balance) As Sum\_Avail\_Balance  From   Account Acc  Group  By Acc.Cust\_Id; |

8.3- SQL AVG

AVG() là hàm tính trung bình trên cột số.

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4 | -- Cú pháp:    SELECT AVG("column\_name")  FROM "table\_name"; |

Ví dụ:

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | -- Tính số tiền trung bình ứng với loại hình gửi tiết kiêm.  Select Avg(Acc.Avail\_Balance)  As Avg\_Avail\_Balance  From   Account Acc  Where  Acc.Product\_Cd = 'SAV';      -- Sử dụng với Group by.  -- Một khách hàng có thể có 1 hoặc nhiều tài khoản.  -- Tính số tiền trung bình mỗi tài khoản ứng với từng khách hàng  -- (tại ngân hàng Branch\_ID = 1)  Select Acc.Cust\_Id      ,Avg(Acc.Avail\_Balance) As Avg\_Avail\_Balance  From   Account Acc  Where  Acc.Open\_Branch\_Id = 1  Group  By Acc.Cust\_Id; |

8.4- SQL MIN

Min là hàm tìm giá trị nhỏ nhất trên cột số.

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4 | -- Cú pháp:    SELECT MIN ("column\_name")  FROM "table\_name"; |

Ví dụ:

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | -- Tìm số tiền gửi tiết kiệm nhỏ nhất.  Select Min(Acc.Avail\_Balance) As Min\_Avail\_Balance  From   Account Acc  Where  Acc.Product\_Cd = 'SAV';      -- Sử dụng với Group by.  -- Một khách hàng có thể có 1 hoặc nhiều tài khoản.  -- Mở tại các chi nhánh khác nhau.  -- Tìm số tiền trong tài khoản nhỏ nhất ứng với từng chi nhánh  Select Acc.Open\_Branch\_Id       ,Min(Acc.Avail\_Balance) As Min\_Avail\_Balance  From   Account Acc  Group  By Acc.Open\_Branch\_Id; |

8.5- SQL MAX

MAX() là hàm tìm giá trị lớn nhất trên một cột số. Cách sử dụng giống MIN, bạn có thể tham khảo các ví dụ tại mục MIN.

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4 | -- Cú pháp:    SELECT MAX("column\_name")  FROM "table\_name"; |

9- SQL Join

Đặt ra một tình huống bạn xem thông tin một nhân viên trong bảng EMPLOYEE. Bạn có thể thấy nhân viên biết ID phòng ban của nhân viên này. Nhưng đó là một con số vô nghĩa. Muốn biết tên phòng ban, bạn phải tra cứu sang bảng DEPARTMENT. Việc nối 2 bảng đó với nhau để có một thông tin đầy đủ được gọi là JOIN.

Có 4 hình thức để Join 2 bảng:

INNER JOIN   (JOIN)

LEFT OUTER JOIN  (LEFT JOIN)

RIGHT OUTER JOIN (RIGHT JOIN)

FULL OUTER JOIN    (OUTER JOIN)

CROSS JOIN

9.1- INNER JOIN (Hoặc JOIN)

Từ khóa INNER JOIN để chọn tất cả các dòng từ hai bảng miễn là có sự ăn khớp dữ liệu giữa các cột trong cả hai bảng.

Cú pháp:

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | -- Cú pháp    SELECT column\_name(s)  FROM table1  INNER JOIN table2  ON table1.column\_name=table2.column\_name;    -- Có thể thay INNER JOIN bởi JOIN  -- Ý nghĩa và kết quả là như nhau.    SELECT column\_name(s)  FROM table1  JOIN table2  ON table1.column\_name=table2.column\_name; |

Ví dụ:

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | -- INNER JOIN 2 bảng EMPLOYEE và DEPARTMENT.    Select Emp.Emp\_Id       ,Emp.First\_Name       ,Emp.Last\_Name       ,Emp.Dept\_Id       ,Dep.Name Dept\_Name  From   Employee Emp  Inner  Join Department Dep  On     Emp.Dept\_Id = Dep.Dept\_Id  Order  By Emp.Emp\_Id; |

Cú pháp riêng của SQLServer:

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | -- Cú pháp này cũng được hỗ trợ bởi Oracle & MySQL  -- Việc INNER JOIN theo cách viết khác của SQLServer thực sự đơn giản.  -- SQLServer đẩy điều kiện JOIN hai bảng xuống WHERE.    Select Emp.Emp\_Id     ,Emp.First\_Name     ,Emp.Last\_Name     ,Emp.Dept\_Id     ,Dep.Name Dept\_Name  From   Employee   Emp     ,Department Dep  Where  Emp.Dept\_Id = Dep.Dept\_Id  Order  By Emp.Emp\_Id; |

9.2- LEFT OUTER JOIN (Hoặc LEFT JOIN)

Từ khóa LEFT OUTER JOIN trả về tất cả các hàng (rows) từ bảng bên trái (table1), với các hàng tương ứng trong bảng bên phải (table2). Chấp nhận cả dữ liệu NULL ở bảng 2 nếu nó không khớp.

Hãy xem hình minh họa dưới đây:

Ví dụ:

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | -- Customer LEFT OUTER JOIN Officer  -- Có thể thay thế LEFT OUTER JOIN bởi LEFT JOIN (Ý nghĩa và kết quả là giống nhau)  Select Cus.Cust\_Id       ,Cus.Fed\_Id       ,Cus.State       ,Ofc.Cust\_Id As Ofc\_Cust\_Id       ,Ofc.Officer\_Id       ,Ofc.Start\_Date       ,Ofc.Title  From   Customer Cus  -- Table1  Left   Outer Join Officer Ofc  -- Table2  On     Cus.Cust\_Id = Ofc.Cust\_Id; |

Kết quả:

9.3- RIGHT OUTER JOIN (Hoặc RIGHT JOIN)

RIGHT OUTER JOIN khá giống với LEFT OUTER JOIN:

9.4- FULL OUTER JOIN (Hoặc OUTER JOIN)

FULL OUTER JOIN là sự kết hợp của LEFT OUTER JOIN và RIGHT OUTER JOIN

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | -- Cú pháp:  (FULL OUTER JOIN)  -- Có thể viết FULL JOIN    SELECT columns  FROM table1  FULL [OUTER] JOIN table2  ON table1.column = table2.column; |

10- Câu truy vấn con (Subquery)

Trong SQLServer, một subquery là một truy vấn trong một truy vấn. Bạn có thể tạo các truy vấn con trong câu lệnh SQL của bạn. Những truy vấn con có thể nằm trong mệnh đề WHERE, mệnh đề FROM, hoặc mệnh đề SELECT.

10.1- Subquery trong mệnh đề Where

Khá thường xuyên, các subquery sẽ được tìm thấy trong mệnh đề WHERE. Những truy vấn con còn được gọi là truy vấn con lồng nhau.

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | Select Acc.Account\_Id       ,Acc.Open\_Date       ,Acc.Product\_Cd       ,Acc.Avail\_Balance  From   Account Acc  Where  Acc.Cust\_Id In        (Select Cus.Cust\_Id From Customer Cus Where Cus.Cust\_Type\_Cd = 'B') |

10.2- Subquery trong mệnh đề From

Một truy vấn phụ cũng có thể được tìm thấy trong mệnh đề FROM. Chúng được gọi là inline views.

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | Select Cus.Cust\_Id       ,Cus.Address       ,Cus.Fed\_Id       ,Acc2.Sum\_Avail\_Balance  From   Customer Cus       , -- Câu Subquery định nghĩa 1 bảng ảo (inline view)        (Select Acc.Cust\_Id               ,Sum(Acc.Avail\_Balance) As Sum\_Avail\_Balance         From   Account Acc         Group  By Acc.Cust\_Id) Acc2  Where  Cus.Cust\_Id = Acc2.Cust\_Id; |

10.3- Subquery trong mệnh đề Select

Một subquery cũng có thể được tìm thấy trong mệnh đề SELECT.

[?](http://o7planning.org/vi/10239/huong-dan-hoc-sql-cho-nguoi-moi-bat-dau-voi-sql-server)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | Select Cus.Cust\_Id       ,Cus.Address       ,Cus.Fed\_Id       ,(Select Sum(Acc.Avail\_Balance)         From   Account Acc         Where  Acc.Cust\_Id = Cus.Cust\_Id) As Sum\_Avail\_Balance  From   Customer Cus; |

Bí quyết để đặt một subquery trong mệnh đề select là subquery phải trả lại một giá trị duy nhất. Đây là lý do tại sao một các hàm tổng hợp như hàm SUM, COUNT, MIN, hoặc MAX thường được sử dụng trong subquery.

1- Transact-SQL là gì

2- Tổng quan về Transact-SQL

3- Bắt đầu với SQL Server Management Studio

4- Các lệnh Transact-SQL cơ bản

4.1- Câu Lệnh rẽ nhánh If-elsif-else

4.2- Vòng lặp WHILE

5- Gán dữ liệu truy vấn vào biến

6- Các kiểu dữ liệu đặc biệt trong T-SQL

6.1- Kiểu dữ liệu TABLE (Dạng tường minh)

6.2- Kiểu dữ liệu TABLE (Dạng không tường minh)

7- Con trỏ (Cursor)

7.1- Con trỏ là gì?

7.2- Khai báo con trỏ

7.3- Ví dụ với Con trỏ

7.4- Ví dụ sử dụng Con trỏ (Khai báo dạng biến)

8- Điều khiển ngoại lệ

9- Hàm (Function)

10- Thủ tục (Procedure)

11- Sử lý giao dịch (Transaction)

11.1- Tại sao cần sử lý giao dịch

11.2- Khai báo và sử dụng giao dịch (Transaction)

12- Trigger

# 1- Transact-SQL là gì

**Transact-SQL** (còn gọi là **T-SQL**) là một ngôn ngữ lập trình **database** hướng thủ tục độc quyền của **Microsoft** sử dụng trong **SQL Server**.   
  
Ngôn ngữ thủ tục được thiết kế để mở rộng khả năng của **SQL** trong khi có khả năng tích hợp tốt với **SQL**. Một số tính năng như các biến địa phương và xử lý chuỗi/dữ liệu được thêm vào. Các tính năng này làm cho ngôn ngữ **Transact-SQL** là **Turing-complete** (\*\*).   
  
Chúng cũng được sử dụng để viết các thủ tục lưu trữ: Một đoạn code nằm trên máy chủ để quản lý các quy tắc kinh doanh phức tạp mà khó hoặc không thể làm nổi với các thao tác tập hợp thuần thúy (pure set-based operations).

*Một hệ thống Turing-Complete có nghĩa là một hệ thống trong đó một chương trình có thể được viết ra và sẽ tìm thấy câu trả lời (mặc dù không có sự bảo đảm về thời gian chạy hoặc bộ nhớ).*

# 2- Tổng quan về Transact-SQL

**T-SQL** tổ chức theo từng khối lệnh, một khối lệnh có thể lồng bên trong một khối lệnh khác, một khối lệnh bắt đầu bởi **BEGIN** và kết thúc bởi **END**, bên trong khối lệnh có nhiều lệnh, và các lệnh ngăn cách nhau bởi dấu chấm phẩy.

Cấu trúc khối lệnh:

[?](http://o7planning.org/vi/10327/huong-dan-lap-trinh-sql-server-transact-sql)

|  |  |
| --- | --- |
| 1  2  3  4 | BEGIN      -- Khai báo biến     -- Các câu lệnh T-SQL  END; |

# 3- Bắt đầu với SQL Server Management Studio

Trong tài liệu này tôi sẽ hướng dẫn bạn lập trình **SQL Server**, trên công cụ trực quan **SQL Server Management Studio**.

Đây là hình ảnh **SQL Server Management Studio** trong khi bạn mới mở nó. Có một số **database** ví dụ có sẵn khi bạn cài đầy đủ **SQLServer**.

Hoặc bạn có thể tạo ***learningsql***, một cơ sở dữ liệu nhỏ được sử dụng trong một vài tài liệu hướng dẫn **SQLServer** trên ***o7planning.org***.

1. [Cơ sở dữ liệu SQL Server mẫu để học SQL](http://o7planning.org/vi/10241/co-so-du-lieu-sql-server-mau-de-hoc-sql)

Nhấn phải chuột vào một **database**, chọn **"New Query"** để mở ra một cửa sổ làm việc với **database** này.

Bạn đã sẵn sàng với lập trình **database** với **SQL Server**.

Dưới đây là một khối lệnh đơn giản, tính tổng 2 số:

[?](http://o7planning.org/vi/10327/huong-dan-lap-trinh-sql-server-transact-sql)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | Begin      -- Khai báo một biến    Declare @v\_Result Int;    -- Khai báo một biến có giá trị 50    Declare @v\_a Int = 50;    -- Khai báo một biến có giá trị 100    Declare @v\_b Int = 100;        -- In ra màn hình Console (Dùng cho lập trình viên).    -- Sử dụng Cast để ép kiểu Int về kiểu chuỗi.    -- Sử dụng toán tử + để nối 2 chuỗi.    Print 'v\_a= ' + Cast(@v\_a as varchar(15));      -- In ra màn hình Console    Print 'v\_b= ' + Cast(@v\_b as varchar(15));      -- Tính tổng    Set @v\_Result = @v\_a + @v\_b;      -- In ra màn hình Console    Print 'v\_Result= ' + Cast(@v\_Result as varchar(15));    End; |

# 4- Các lệnh Transact-SQL cơ bản

Ở đây tôi giới thiệu tổng quan về các lệnh cơ bản của **T-SQL**. Bạn sẽ hiểu hơn về nó thông qua các ví dụ ở các phần tiếp theo.

## 4.1- Câu Lệnh rẽ nhánh If-elsif-else

Cú pháp:

[?](http://o7planning.org/vi/10327/huong-dan-lap-trinh-sql-server-transact-sql)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | IF <điều kiện 1>        Khối lệnh 1;  [ELSE IF <điều kiện 2>        Khối lệnh 2;  ]  ....  [ELSE        Khối lệnh n + 1;  ] |

Ví dụ:

Else\_If\_Example

[?](http://o7planning.org/vi/10327/huong-dan-lap-trinh-sql-server-transact-sql)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | BEGIN    -- Khai báo một biến  DECLARE @v\_Option integer;  DECLARE @v\_Action varchar(30);    SET @v\_Option = 2;    IF @v\_Option = 1    SET @v\_Action = 'Run';  ELSE IF @v\_Option = 2      BEGIN        PRINT 'In block else if @v\_Option = 2';        SET @v\_Action = 'Backup';      END;  ELSE IF @v\_Option = 3    SET @v\_Action = 'Stop';  ELSE    SET @v\_Action = 'Invalid';      -- Ghi ra log  PRINT '@v\_Action= ' + @v\_Action;    END; |

Kết quả chạy ví dụ:

## 4.2- Vòng lặp WHILE

Cú pháp:

[?](http://o7planning.org/vi/10327/huong-dan-lap-trinh-sql-server-transact-sql)

|  |  |
| --- | --- |
| 1  2  3  4 | WHILE condition  BEGIN     -- ...statements...  END; |

Trong vòng lặp **WHILE** bạn có thể sử dụng **BREAK** để thoát ra khỏi vòng lặp.   
Sử dụng lệnh **CONTINUE** để bỏ qua các dòng lệnh trong khối **WHILE** và ở bên dưới nó, để tiếp tục một vòng lặp mới.

While\_Example1

[?](http://o7planning.org/vi/10327/huong-dan-lap-trinh-sql-server-transact-sql)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | BEGIN     -- Khai báo 2 biến x và y.   DECLARE @x integer = 0;   DECLARE @y integer = 10;     -- Bước   DECLARE @step integer = 0;     -- Trong khi @x < @y   WHILE (@x < @y)   BEGIN       SET @step = @step + 1;       -- Mỗi lần vòng lặp chạy giá trị của x tăng lên 1     SET @x = @x + 1;     -- Mỗi lần vòng lặp chạy giá trị của y giảm đi 2     SET @y = @y - 2;       PRINT 'Step =' + CAST(@step AS varchar(10));     PRINT '@x =' + CAST(@x AS varchar(10)) + ' / @y = ' + CAST(@y AS varchar(10));     END;     -- Ghi ra log   PRINT 'x,y = ' + CAST(@x AS varchar(10)) + ', ' + CAST(@y AS varchar(10));    END; |

Kết quả chạy ví dụ:

**BREAK** là lệnh cho phép thoát ra khỏi vòng lặp, dưới đây là ví dụ:

While\_Example2

[?](http://o7planning.org/vi/10327/huong-dan-lap-trinh-sql-server-transact-sql)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34 | BEGIN    -- Khai báo 2 biến x và y.  DECLARE @x integer = 0;  DECLARE @y integer = 10;    -- Bước  DECLARE @step integer = 0;    -- Trong khi @x < @y  WHILE (@x < @y)  BEGIN      SET @step = @step + 1;      -- Mỗi lần vòng lặp chạy giá trị của x tăng lên 1    SET @x = @x + 1;    -- Mỗi lần vòng lặp chạy giá trị của y giảm đi 2    SET @y = @y - 2;      PRINT 'Step =' + CAST(@step AS varchar(10));    PRINT '@x =' + CAST(@x AS varchar(10)) + ' / @y = ' + CAST(@y AS varchar(10));      -- Nếu @x > 2 thì thoát ra khỏi vòng  lặp    -- (Mặc dù điều kiện trong WHILE vẫn đúng).    IF @x > 2      BREAK;    END;    -- Ghi ra log  PRINT 'x,y = ' + CAST(@x AS varchar(10)) + ', ' + CAST(@y AS varchar(10));    END; |

Kết quả chạy ví dụ:

Lệnh **CONTINUE** cho phép bỏ qua các câu lệnh bên dưới nó (vẫn trong vòng lặp) để tiếp tục vòng lặp mới.

While\_Example3

[?](http://o7planning.org/vi/10327/huong-dan-lap-trinh-sql-server-transact-sql)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35 | BEGIN    -- Khai báo 2 biến x và y.  DECLARE @x integer = 0;  DECLARE @y integer = 10;    -- Bước  DECLARE @step integer = 0;    -- Trong khi @x < @y  WHILE (@x < @y)  BEGIN      SET @step = @step + 1;      -- Mỗi lần vòng lặp chạy giá trị của x tăng lên 1    SET @x = @x + 1;    -- Mỗi lần vòng lặp chạy giá trị của y giảm đi 2    SET @y = @y - 2;      -- Nếu @x < 3 thì bỏ qua các dòng lệnh bên dưới    -- Tiếp tục vòng lặp mới.    IF @x < 3      CONTINUE;      -- Nếu @x < 3 các dòng lệnh bên dưới CONTINUE sẽ không được chạy.    PRINT 'Step =' + CAST(@step AS varchar(10));    PRINT '@x =' + CAST(@x AS varchar(10)) + ' / @y = ' + CAST(@y AS varchar(10));    END;    -- Ghi ra log  PRINT 'x,y = ' + CAST(@x AS varchar(10)) + ', ' + CAST(@y AS varchar(10));    END; |

# 5- Gán dữ liệu truy vấn vào biến

Các biến có thể được gán giá trị từ một câu truy vấn. Dưới đây là một ví dụ minh họa:

Assign\_Value\_Example

[?](http://o7planning.org/vi/10327/huong-dan-lap-trinh-sql-server-transact-sql)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | BEGIN    -- Khai báo một biến @v\_Emp\_ID  DECLARE @v\_Emp\_ID integer = 1;    DECLARE @v\_First\_Name varchar(30);  DECLARE @v\_Last\_Name varchar(30);  DECLARE @v\_Dept\_ID integer;    -- Gán giá trị cho các biến lấy từ câu lệnh Select.  SELECT   @v\_First\_Name = emp.First\_Name,   @v\_Last\_Name = emp.Last\_Name,   @v\_Dept\_Id = emp.Dept\_Id  FROM Employee Emp  WHERE Emp.Emp\_ID = @v\_Emp\_Id;    -- In ra các giá trị:  PRINT '@v\_First\_Name = ' + @v\_First\_Name;  PRINT '@v\_Last\_Name = ' + @v\_Last\_Name;  PRINT '@v\_Dept\_Id = ' + CAST(@v\_Dept\_ID AS varchar(15));    END; |

Kết quả chạy ví dụ:

# 6- Các kiểu dữ liệu đặc biệt trong T-SQL

## 6.1- Kiểu dữ liệu TABLE (Dạng tường minh)

**T-SQL** cho phép bạn khai báo một biến có kiểu dữ liệu **TABLE**.

Cú pháp:

[?](http://o7planning.org/vi/10327/huong-dan-lap-trinh-sql-server-transact-sql)

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | -- Định nghĩa một biến có kiểu TABLE.  -- Chú ý: Các giàng buộc cũng có thể tham gia vào khai báo biến kiểu TABLE (Xem trong ví dụ).  Declare @v\_variable\_name  TABLE  (    Column1 DataType1,    Column2 DataType2  ); |

Ví dụ:

[?](http://o7planning.org/vi/10327/huong-dan-lap-trinh-sql-server-transact-sql)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | -- Ví dụ khai báo một biến có kiểu TABLE.  Declare @v\_Table TABLE  (  First\_Name Varchar(30),  Last\_Name Varchar(30),  Dept\_ID Integer,  Salary Float  );      -- Các giàng buộc cũng có thể tham gia vào trong định nghĩa biến kiểu TABLE:  -- Ví dụ:  Declare @v\_table TABLE  (    Product\_ID Integer IDENTITY(1,1) PRIMARY KEY,    Product\_Name  DataType2 NOT NULL Default ('Unknown'),    Price Money CHECK (Price < 10.0)  ); |

Ví dụ: Insert dữ liệu vào biến có kiểu **TABLE**.

[?](http://o7planning.org/vi/10327/huong-dan-lap-trinh-sql-server-transact-sql)

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | Insert Into      @v\_Table (First\_Name, Last\_Name, Dept\_ID, Salary)  Select Emp.First\_Name, Emp.Last\_Name, Emp.Dept\_Id, 1000  From      Employee Emp  Where Emp.Emp\_ID < 4; |

Bạn cũng có thể **Update** trên biến có kiểu **TABLE**:

[?](http://o7planning.org/vi/10327/huong-dan-lap-trinh-sql-server-transact-sql)

|  |  |
| --- | --- |
| 1  2  3  4 | Update     @v\_Table  Set Salary = Salary + 100  Where Dept\_Id = 10; |

**Delete** trên biến có kiểu **TABLE**:

[?](http://o7planning.org/vi/10327/huong-dan-lap-trinh-sql-server-transact-sql)

|  |  |
| --- | --- |
| 1 | Delete From @v\_Table Where Dept\_ID = 10; |

**Query** dữ liệu trên biến có kiểu **TABLE**:

[?](http://o7planning.org/vi/10327/huong-dan-lap-trinh-sql-server-transact-sql)

|  |  |
| --- | --- |
| 1  2  3 | Select \* from @v\_Table  Where Dept\_ID = 10  Order by First\_Name; |

Ví dụ:

[?](http://o7planning.org/vi/10327/huong-dan-lap-trinh-sql-server-transact-sql)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | BEGIN     DECLARE @v\_Emp\_ID integer = 1;     -- Khai báo một biến kiểu TABLE.   DECLARE @v\_Table TABLE (     First\_Name varchar(30),     Last\_Name varchar(30),     Dept\_Id integer,     Salary float DEFAULT 1000   );     -- Sử dụng INSERT INTO để trèn dữ liệu vào @v\_Table.   INSERT INTO @v\_Table (First\_name, Last\_Name, Dept\_ID)     SELECT       emp.First\_Name,       emp.Last\_Name,       emp.Dept\_Id     FROM Employee Emp     WHERE Emp.Emp\_ID < 4;     -- Update @v\_Table   UPDATE @v\_Table   SET Salary = Salary + 100   WHERE First\_name = 'Susan';     -- Duyệt các giá trị trên @v\_Table.   SELECT     \*   FROM @v\_Table;    END; |

Kết quả chạy ví dụ:

## 6.2- Kiểu dữ liệu TABLE (Dạng không tường minh)

**T-SQL** cho phép bạn khai báo một biến kiểu **TABLE** không tường minh. Tên biến bắt đầu bởi **#**.

Table\_Example

[?](http://o7planning.org/vi/10327/huong-dan-lap-trinh-sql-server-transact-sql)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | BEGIN    -- Sử dụng SELECT INTO để trèn dữ liệu vào #v\_My\_Table.  SELECT   emp.First\_Name,   emp.Last\_Name,   emp.Dept\_Id,   1000 Salary INTO #v\_My\_Table  FROM Employee Emp  WHERE Emp.Emp\_ID < 4;    -- Update #v\_My\_Table  UPDATE #v\_My\_Table  SET Salary = Salary + 100  WHERE First\_name = 'Susan';    -- Duyệt các giá trị trên #v\_My\_Table.  SELECT   \*  FROM #v\_My\_Table;    END; |

Kết quả chạy ví dụ:

# 7- Con trỏ (Cursor)

## 7.1- Con trỏ là gì?

**Cursor** là kiểu biến có cấu trúc, cho phép bạn xử lý dữ liệu gồm nhiều dòng. Số dòng phụ thuộc vào câu lệnh truy vấn dữ liệu. Trong quá trình xử lý, bạn có thể thao tác với **cursor** thông qua từng dòng dữ liệu. Dòng dữ liệu này được định vị bởi một con trỏ. Với việc dịch chuyển con trỏ, bạn có thể truy cập tất cả các dòng dữ liệu.

## 7.2- Khai báo con trỏ

Cú pháp:

[?](http://o7planning.org/vi/10327/huong-dan-lap-trinh-sql-server-transact-sql)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | -- ISO Syntax  DECLARE cursor\_name [ INSENSITIVE ] [ SCROLL ] CURSOR      FOR select\_statement      [ FOR { READ ONLY | UPDATE [ OF column\_name [ ,...n ] ] } ]  [;]    -- Transact-SQL Extended Syntax  DECLARE cursor\_name CURSOR [ LOCAL | GLOBAL ]      [ FORWARD\_ONLY | SCROLL ]      [ STATIC | KEYSET | DYNAMIC | FAST\_FORWARD ]      [ READ\_ONLY | SCROLL\_LOCKS | OPTIMISTIC ]      [ TYPE\_WARNING ]      FOR select\_statement      [ FOR UPDATE [ OF column\_name [ ,...n ] ] ]  [;] |

## 7.3- Ví dụ với Con trỏ

Cursor\_Example

[?](http://o7planning.org/vi/10327/huong-dan-lap-trinh-sql-server-transact-sql)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45 | USE learningsql;    BEGIN    --  -- Khai báo biến:  DECLARE @v\_Emp\_ID integer;  DECLARE @v\_First\_Name varchar(50);  DECLARE @v\_Last\_Name varchar(50);    DECLARE @v\_Count integer;    -- Khai báo một con trỏ (CURSOR).  DECLARE My\_Cursor CURSOR FOR  SELECT   Emp.EMP\_ID,   Emp.FIRST\_NAME,   Emp.LAST\_NAME  FROM Employee Emp  WHERE Emp.EMP\_ID < 3;        -- Mở Cursor  OPEN My\_Cursor;    -- Di chuyển con trỏ từ đến dòng đầu tiên.  -- Và gán các giá trị cột vào các biến.  FETCH NEXT FROM My\_Cursor INTO @v\_Emp\_ID, @v\_First\_Name, @v\_Last\_Name;    -- Trong trường hợp còn bản ghi @@FETCH\_STATUS = 0.  WHILE @@FETCH\_STATUS = 0  BEGIN   PRINT 'First Name = '+ @v\_First\_Name+' / Last Name = '+ @v\_Last\_Name;     -- Di chuyển đến bản ghi tiếp theo.   -- Và gán các giá trị cột vào các biến.   FETCH NEXT FROM My\_Cursor INTO @v\_Emp\_ID, @v\_First\_Name, @v\_Last\_Name;  END    -- Đóng Cursor.  CLOSE My\_Cursor;  DEALLOCATE My\_Cursor;    END; |

Kết quả chạy ví dụ:

## 7.4- Ví dụ sử dụng Con trỏ (Khai báo dạng biến)

Cursor\_Example2

[?](http://o7planning.org/vi/10327/huong-dan-lap-trinh-sql-server-transact-sql)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45 | USE learningsql;    BEGIN    --  -- Khai báo biến:  DECLARE @v\_Emp\_ID integer;  DECLARE @v\_First\_Name varchar(50);  DECLARE @v\_Last\_Name varchar(50);    -- Khai báo một biến kiểu con trỏ.  DECLARE @My\_Cursor CURSOR;    -- Sét câu lệnh truy vấn cho con trỏ.  Set @My\_Cursor = CURSOR FOR  SELECT   Emp.EMP\_ID,   Emp.FIRST\_NAME,   Emp.LAST\_NAME  FROM Employee Emp  WHERE Emp.EMP\_ID < 3;      -- Mở Cursor  OPEN @My\_Cursor;    -- Di chuyển con trỏ đến dòng đầu tiên.  -- Và gán các giá trị cột vào các biến.  FETCH NEXT FROM @My\_Cursor INTO @v\_Emp\_ID, @v\_First\_Name, @v\_Last\_Name;    -- Trong trường hợp có bản ghi @@FETCH\_STATUS = 0.  WHILE @@FETCH\_STATUS = 0  BEGIN   PRINT 'First Name = '+ @v\_First\_Name+' / Last Name = '+ @v\_Last\_Name;     -- Di chuyển đến bản ghi tiếp theo.   -- Và gán các giá trị cột vào các biến.   FETCH NEXT FROM @My\_Cursor INTO @v\_Emp\_ID, @v\_First\_Name, @v\_Last\_Name;  END    -- Đóng Cursor.  CLOSE @My\_Cursor;  DEALLOCATE @My\_Cursor;    END; |

Kết quả chạy ví dụ:

# 8- Điều khiển ngoại lệ

Sponsored Content

[Recommended by](http://www.outbrain.com/what-is/default/en)

Khi lập trình **T-SQL** có thể có những lỗi xuất hiện trong Code của bạn, chẳng hạn như lỗi chia cho 0. Hoặc lỗi khi bạn trèn một bản ghi trùng lặp khóa chính, ... Bạn cần phải xử lý các tình huống này.

Hãy xem một ví dụ đơn giản, xử lý lỗi chia cho 0.

TryCatch\_Example

[?](http://o7planning.org/vi/10327/huong-dan-lap-trinh-sql-server-transact-sql)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47 | USE learningsql;    BEGIN    --  -- 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 bẫy lỗi.  -- Nếu lỗi xẩy ra trong khối này  -- nó sẽ nhẩy 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 xẩy ra tại đây.    SET @v\_c = @v\_a / @v\_b;      -- Dòng bên dưới này sẽ không được chạy.    -- Chương trình nhẩy 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.  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; |

Kết quả chạy ví dụ:

Thông tin lỗi:

|  |  |
| --- | --- |
| **Hàm** | **Mô tả** |
| ERROR\_NUMBER() | Trả về mã lỗi. |
| ERROR\_MESSAGE() | Trả về văn bản đầy đủ của các thông báo lỗi. Các văn bản bao gồm các giá trị đã cung cấp cho các tham số, chẳng hạn như độ dài, tên đối tượng, hoặc thời gian. |
| ERROR\_SEVERITY() | Trả về mức độ nghiêm trọng của lỗi. |
| ERROR\_STATE() | Trả về trạng thái lỗi |
| ERROR\_LINE() | Trả về số vị trí dòng code gây lỗi. |
| ERROR\_PROCEDURE() | Trả về tên của **stored procedure** hoặc **trigger** nơi mà lỗi phát ra. |

# 9- Hàm (Function)

Giống như **procedure** (thủ tục), **function** (hàm ) là nhóm các lệnh **T-SQL** thực hiện chức năng nào đó. Khác với thủ tục, các hàm sẽ trả về một giá trị ngay tại lời gọi nó.   
  
Hàm cũng có thể được lưu giữ ngay trên **database** dưới dạng **Store procedure**.

Cú pháp tạo **function** (Hàm).

[?](http://o7planning.org/vi/10327/huong-dan-lap-trinh-sql-server-transact-sql)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | -- function\_name:  Tên hàm  -- argument:  Tên tham số  -- mode:  INPUT, OUTPUT,  hoặc không cần viết.  -- datatype:  Kiểu dữ liệu của tham số    CREATE FUNCTION <function\_name>             (                [                 @argument1 datatype1 [mode1] ,                 @argument2  datatype2 [mode2],                 ...                 ]              )             RETURNS  datatype  AS  BEGIN   -- Khai báo biến sử dụng   -- Code nội dung của hàm   -- Trả về giá trị của hàm.  END; |

Ví dụ:

[?](http://o7planning.org/vi/10327/huong-dan-lap-trinh-sql-server-transact-sql)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | -- Một hàm có tham số:  CREATE FUNCTION Sum\_Ab(a Integer, b Integer)  RETURNS Integer  AS  Begin  return a + b;  End;    -- Một hàm không tham số:  CREATE FUNCTION Get\_Current\_Datetime()  RETURNS Date  AS  Begin  return CURRENT\_TIMESTAMP;  End; |

Hủy **Function** (Drop function):

[?](http://o7planning.org/vi/10327/huong-dan-lap-trinh-sql-server-transact-sql)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | -- Hủy Function    DROP FUNCTION <function\_name>;    -- Ví dụ:    DROP FUNCTION My\_Function; |

### Ví dụ tạo một hàm.

Đây là một ví dụ tạo một **function** (hàm) đầu tiên của bạn với **SQL Server**:

1. Tạo một hàm (Function)
2. Biên dịch hàm này
3. Chạy hàm

[?](http://o7planning.org/vi/10327/huong-dan-lap-trinh-sql-server-transact-sql)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | -- Kiểm tra xem Hàm này đã tồn tại chưa.  -- Nếu đã tồn tại cần phải hủy để có thể tạo mới.  IF OBJECT\_ID(N'dbo.My\_Sum', N'FN') IS NOT NULL   DROP FUNCTION My\_Sum;  GO    CREATE FUNCTION My\_Sum (@p\_a float, @p\_b float)  RETURNS float  AS  BEGIN     -- Khai báo một biến Float   DECLARE @v\_C float;     -- Sét giá trị cho biến v\_C   SET @V\_C = @p\_A + @p\_B;     -- Giá trị trả về của hàm.   RETURN @v\_C;    END; |

Nhấn biểu tượng để biên dịch hàm.

Hàm bạn vừa tạo ra ở trên là một hàm đơn giản trả về một giá trị vô hướng (Scalar-value). Bạn có thể nhìn thấy nó đã được tạo ra trên **SQLServer Management Studio**:

Bạn có thể test hàm bằng cách nhấn phải chuột vào hàm, chọn:

1. **Script function as -> SELECT to -> New Query Editor Window**

Một cửa sổ test được mở ra, bạn có thể sửa đổi các tham số truyền vào:

Sửa đổi các giá trị tham số truyền vào và nhấn nút thực thi để test.

Các hàm có thể tham gia vào câu lệnh **SELECT**.

[?](http://o7planning.org/vi/10327/huong-dan-lap-trinh-sql-server-transact-sql)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | SELECT    acc.account\_id,    acc.cust\_id,    acc.avail\_balance,    acc.pending\_balance,    dbo.MY\_SUM(acc.avail\_balance, acc.pending\_balance) balance  FROM account acc; |

Kết quả chạy câu lệnh SQL trên:

# 10- Thủ tục (Procedure)

Một nhóm các lệnh **T-SQL** thực hiện chức năng nào đó có thể được gom lại trong một thủ tục (procedure) nhằm làm tăng khả năng xử lý,khả năng sử dụng chung, tăng tính bảo mật và an toàn dữ liệu, tiện ích trong phát triển.   
  
Thủ tục có thể được lưu giữ ngay trong **database** như một đối tượng của **database**, sẵn sàng cho việc tái sử dụng. Thủ tục lúc này được gọi là **Store procedure**. Với các **Store procedure**, ngay khi lưu giữ **Store procedure**, chúng đã được biên dịch thành dạng **p-code** vì thế có thể nâng cao khả năng thực hiện.

Thủ tục không trả về giá trị trực tiếp như hàm ( **function**). Tuy nhiên nó có thể có 0 hoặc nhiều tham số đầu ra.

Cú pháp tạo một thủ tục:

[?](http://o7planning.org/vi/10327/huong-dan-lap-trinh-sql-server-transact-sql)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | -- procedure\_name:  Tên thủ tục  -- argument:  Tên tham số  -- mode:  Loại tham số: INPUT hoặc OUTPUT, mặc định là INPUT  -- datatype:  Kiểu dữ liệu của tham số  -- Chú ý: Với thủ tục các tham số có thể đặt trong dấu () hoặc không cần thiết.    CREATE PROCEDURE <procedure\_name>       [        argument1  datatype1 [mode1]  ,        argument2   datatype2 [mode2]  ,       ...        ]  AS  BEGIN   -- Khai báo biến sử dụng   -- Nội dung của thủ tục.  END;    -- Hoặc:    CREATE PROCEDURE <procedure\_name>     (       [        argument1  datatype1 [mode1]  ,        argument2   datatype2 [mode2]  ,       ...        ]     )  AS  BEGIN   -- Khai báo biến sử dụng   -- Nội dung của thủ tục.  END; |

Ví dụ:

[?](http://o7planning.org/vi/10327/huong-dan-lap-trinh-sql-server-transact-sql)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | -- Ví dụ một thủ tục không tham số.  CREATE Procedure Do\_Something  AS  Begin       -- Khai báo biến tại đây.       Declare @v\_a Integer;        -- Làm gì đó tại đây.        -- ....  End;    -- Ví dụ một thủ tục có tham số  --  Có một tham số đầu vào và một tham số đầu ra.    CREATE Procedure Do\_Something (@p\_Param1 Varchar(20),                               @v\_Param2 Varchar(50) OUTPUT )  AS  Begin    -- Khai báo biến tại đây.    Declare @v\_a Integer;      -- Làm gì đó tại đây.    -- ...  End; |

Hủy thủ tục (Drop procedure):

[?](http://o7planning.org/vi/10327/huong-dan-lap-trinh-sql-server-transact-sql)

|  |  |
| --- | --- |
| 1  2  3 | -- Hủy một thủ tục:    DROP PROCEDURE <Procedure\_Name> |

Các bước thực hiện một thủ tục:

### Ví dụ tạo một thủ tục:

Ở đây tôi tạo một thủ tục đơn giản, với tham số truyền vào là **@p\_Emp\_ID** và có 3 tham số đầu ra, **@v\_First\_Name**, **@v\_Last\_Name**, **@v\_Dep\_ID**.

Get\_Employee\_Infos

[?](http://o7planning.org/vi/10327/huong-dan-lap-trinh-sql-server-transact-sql)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37 | -- Huỷ procedure Get\_Employee\_Infos nếu nó đã tồn tại.  -- (Để cho phép tạo lại)  IF OBJECT\_ID(N'dbo.Get\_Employee\_Infos', N'P') IS NOT NULL  DROP PROCEDURE Get\_Employee\_Infos;  GO    -- Thủ tục truyền vào p\_Emp\_Id  -- Và trả về v\_First\_Name, v\_Last\_Name, v\_Dept\_Id.  CREATE PROCEDURE Get\_Employee\_Infos (@p\_Emp\_Id integer        , @v\_First\_Name varchar(50) OUTPUT        , @v\_Last\_Name varchar(50) OUTPUT        , @v\_Dept\_Id integer OUTPUT)  AS  BEGIN  -- Sử dụng lệnh Print để in ra một chuỗi (Dành cho lập trình viên).  -- Sử dụng Cast để ép kiểu số Integer về dạng chuỗi (Varchar).  -- Sử dụng toán tử + để nối hai chuỗi.  PRINT 'Parameter @p\_Emp\_Id = ' + CAST(@p\_Emp\_ID AS varchar(15));  --  -- Query dữ liệu từ bảng và gán giá trị vào các biến.  --  SELECT    @v\_First\_Name = Emp.First\_Name,    @v\_Last\_Name = Emp.Last\_Name,    @v\_Dept\_Id = Emp.Dept\_Id  FROM Employee Emp  WHERE Emp.Emp\_Id = @p\_Emp\_Id;    --  -- Log dành cho người lập trình  --  PRINT 'Found Record!';  PRINT ' @v\_First\_Name= ' + @v\_First\_Name;  PRINT ' @v\_Last\_Name= ' + @v\_Last\_Name;  PRINT ' @v\_Dept\_Id= ' + CAST(@v\_Dept\_Id AS varchar(15));    END; |

Nhấn vào biểu tượng để biên dịch thủ tục.

Thủ tục đã được tạo ra, bạn có thể nhìn thấy trên **SQL Server Management Studio**:

### Test thủ tục

Trong lập trình việc test một thủ tục và dò tìm lỗi là vấn đề rất quan trọng. Nhấn phải chuột vào thủ tục muốn test, chọn:

1. **Script stored Procedure as -> EXECUTE to -> New Query Editor Window**

Kịch bản test được tạo ra (Mặc định) như hình minh họa dưới đây:

Sét đặt giá trị cho các tham số đầu vào:

Nhấn nút thực thi thủ tục:

# 11- Sử lý giao dịch (Transaction)

## 11.1- Tại sao cần sử lý giao dịch

Giao dịch (Transaction) là một khái niệm quan trọng trong SQL. Hãy xem một tình huống:   
  
Một giao dịch trong ngân hàng, người A chuyển cho người B một khoản tiền 100$, khi đó trong Database xẩy ra 2 thao tác:

1. Trừ tiền của người A đi 100$
2. Cộng tiền vào cho người B 100$.

Điều gì sẽ xẩy ra nếu chỉ có 1 thao tác thành công?

Xem một ví dụ khác:

Khi bạn thêm một sinh viên vào một lớp học bạn cập nhập lại sĩ số của lớp học. Nếu việc trèn thông tin sinh viên không thành công mà sĩ số lại được cộng thêm 1, tính toàn vẹn của dữ liệu bị hỏng.

[?](http://o7planning.org/vi/10327/huong-dan-lap-trinh-sql-server-transact-sql)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | -- Insert một Sinh viên vào bảng Student.  Insert into Student (Studen\_Id, Student\_Name, Class\_ID)  values (100, 'Tom', 1);    -- Cập nhập sĩ số của lớp học.  Update Class\_Table  set Student\_Count = Student\_Count + 1  Where Class\_Id = 1; |

Giao dịch được coi là thành công nếu tất cả các đơn vị lệnh thành công. Ngược lại một trong các đơn vị lệnh bị lỗi, toàn bộ giao dịch cần phải được trả về (rollback) trạng thái ban đầu.

## 11.2- Khai báo và sử dụng giao dịch (Transaction)

Các lệnh liên quan:

1. Bắt đầu **transaction**:
   1. **begin tran** / **begin transaction**
2. Hoàn tất transaction:
   1. **commit**/ **commit tran** / **commit transaction**
3. Quay lui transaction (Rollback transaction):
   1. **rollback** / **rollback tran** / **rollback transaction**
4. Đánh dấu một **savepoint** trong transaction: **save transaction tên\_của\_savepoint**
5. Biến **@@trancount**: cho biết số transaction hiện đang thực hiện (chưa được kết thúc với **rollback** hay **commit**) trong **connection** hiện hành.

Ghi chú:

1. Lệnh **rollback tran + tên\_của\_savepoint** có tác dụng quay lui (rollback) giao dịch đến vị trí đặt **savepoint** tương ứng (không có tác dụng kết thúc transaction), các khóa (locks) đã được thiết lập khi thực hiện các thao tác nằm trong phần bị **rollback** sẽ được mở ra (unlock).
2. Khi khai báo **transaction** tường minh, phải đảm bảo rằng sau đó nó được **rollback** hoặc **commit** tường minh, nếu không, **transaction** sẽ tiếp tục tồn tại và chiếm giữ tài nguyên, ngăn trở sự thực hiện của các **transaction** khác.
3. Lệnh **rollback** chỉ có tác dụng quay lui các giao dịch trên cơ sở dữ liệu (**insert**, **delete**, **update**). Các câu lệnh khác, chẳng hạn lệnh gán, sẽ không bị ảnh hưởng bởi lệnh **rollback**.

Ví dụ:

Transaction\_Example1

[?](http://o7planning.org/vi/10327/huong-dan-lap-trinh-sql-server-transact-sql)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64 | BEGIN    -- Trong ví dụ này các tài khoản Account\_ID = 1, 2 thực sự tồn tại trong DB  -- Trong thực tế bạn có thể viết các câu lệnh kiểm tra trước khi bắt đầu giao dịch.  --  -- Tài khoản người A (Đã đảm bảo tồn tại trong DB)  DECLARE @Account\_Id\_A integer = 1;  -- Tài khoản người B (Đã đảm bảo tồn tại trong DB)  DECLARE @Account\_Id\_B integer = 2;  -- Số tiền chuyển:  DECLARE @Amount float = 10;  -- Giao dịch tại ngân hàng:  DECLARE @Execute\_Branch\_Id integer = 1;    -- Ghi ra số Transaction hiện thời.  -- Thực tế lúc này chưa có giao dịch nào.  PRINT '@@TranCount = ' + CAST(@@Trancount AS varchar(5));    PRINT 'Begin transaction';    -- Bắt đầu giao dịch  BEGIN TRAN;     -- Bẫy lỗi.   BEGIN TRY     --     -- Trừ tiền trong tài khoản người A đi 10$ (Account\_ID = 1)     UPDATE Account     SET AVAIL\_BALANCE = AVAIL\_BALANCE - @Amount     WHERE Account\_Id = @Account\_Id\_A;     --     -- Ghi thông tin thời điểm giao dịch vào bảng Acc\_Transaction.     INSERT INTO ACC\_TRANSACTION (TXN\_DATE, FUNDS\_AVAIL\_DATE, TXN\_TYPE\_CD,      ACCOUNT\_ID, AMOUNT, EXECUTION\_BRANCH\_ID)       VALUES (CURRENT\_TIMESTAMP, CURRENT\_TIMESTAMP, 'CDT',        @Account\_Id\_A, -@Amount, @Execute\_Branch\_Id);     --     -- Cộng tiền vào tài khoản người B thêm 10$     UPDATE Account     SET AVAIL\_BALANCE = AVAIL\_BALANCE + @Amount     WHERE Account\_Id = @Account\_Id\_B;     --     -- Ghi thông tin thời điểm giao dịch vào bảng Acc\_Transaction.     INSERT INTO ACC\_TRANSACTION (TXN\_DATE, FUNDS\_AVAIL\_DATE, TXN\_TYPE\_CD,       ACCOUNT\_ID, AMOUNT, EXECUTION\_BRANCH\_ID)       VALUES (CURRENT\_TIMESTAMP, CURRENT\_TIMESTAMP, 'CDT', @Account\_Id\_B,        @Amount, @Execute\_Branch\_Id);     -- Hoàn thành giao dịch     IF @@Trancount > 0       PRINT 'Commit Transaction';   COMMIT TRAN;    END TRY  -- Nếu có lỗi khối Catch sẽ được chạy.  BEGIN CATCH   PRINT 'Error: ' + ERROR\_MESSAGE();   PRINT 'Error --> Rollback Transaction';   IF @@Trancount > 0     ROLLBACK TRAN;  END CATCH;        END; |

Kết quả chạy ví dụ:

# 12- Trigger

Tài liệu về Trigger được tách riêng, bạn có thể xem hướng dẫn tại: