**View:** dùng để lưu thông tin 1 đối tượng

**Store routines:** dùng để lưu các nhiệm vụ (hành động) hay dùng

Bao gồm 2 loại là Store procedure và function

1. **Store procedure**

* Cú pháp
  + Create store procedure

DELIMITER $$ (hoặc DELIMITER //)

CREATE PROCEDURE *procedure\_name* (IN in\_param\_name\_1 Datatype, OUT out\_param\_name\_2 Datatype)

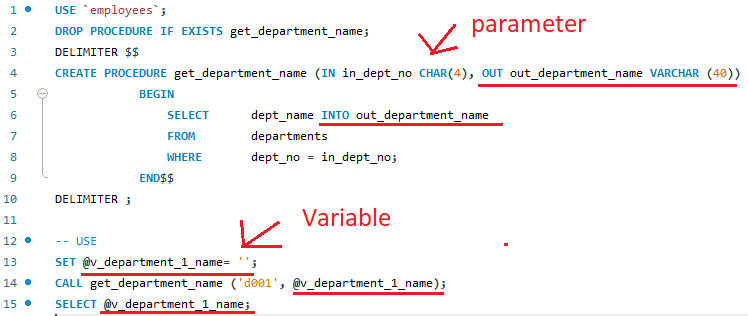
BEGIN

SELECT …;

END$$

DELIMITER ;

Nếu có chứa parameter OUT thì phải sử dụng SELECT…INTO để gán giá trị vào parameter OUT



Phân biệt Parameter và Variable

Parameter: là tham số mà mình phải truyền vào trong procedure

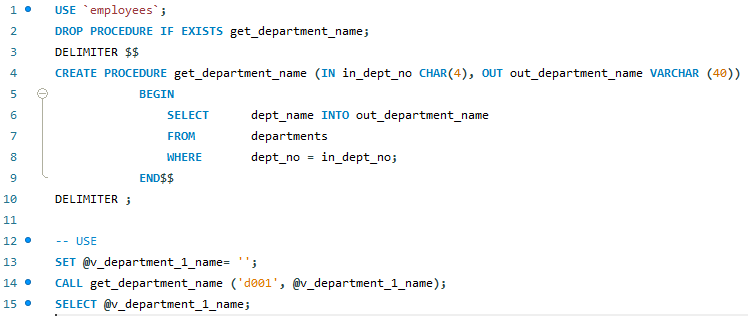
Variable: là biến mà hứng kết quả

* + Drop store procedure

DROP PROCEDURE IF EXISTS *procedure\_name();*

* + Call
    - CALL *procedure\_name();*

VD:



1. **Funtion**

**Built-in function: aggregate funtion, datetime function, …**

* Cú pháp
  + CREATE FUNCTION

DELIMITER $$ (hoặc DELIMITER //)

CREATE FUNCTION function\_name (param\_name\_1 Datatype) RETURNS Datatype

BEGIN

DECLARE variable\_name datatype;

SELECT …;

RETURN variable\_name;

END$$

DELIMITER ;

* + SELECT data

SELECT *function\_name(*param\_name\_1*)*;

* + DROP FUNCTION

DROP FUNCTION IF EXISTS *function\_name;*

1. **Parameter**

Có thể dùng để lưu kết quả của store procedure (OUT parameter)

Create 1 variable

SET @v\_name\_variable = value;

Hiển thị value của variable

SELECT @v\_name\_variable;

1. **Khác nhau giữa Function và Procedure**

|  |  |
| --- | --- |
| **Function** | **Procedure** |
| Chỉ trả về 1 out parameter | Có thể trả về nhiều out parameter |
| Dùng câu lệnh SELECT để sử dụng function 🡺 có thể xử lý out parameter cho các logic tiếp theo | Dùng câu lệnh CALL để sử dụng store procedure |
| Không sử dụng cho Insert, Update, Delete | Sử dụng cho Insert, Update, Delete, Select |
| Không thể return ra table | Không có out parameter là table |

1. **Variable**
   1. **Normal**

DECLARE name\_variable Datatype

* 1. **Table**
     1. **Create**

CREATE TEMPORARY TABLE table\_name(

column\_1\_definition,

column\_2\_definition,

...,

table\_constraints

);

* + 1. **Insert data**

INSERT INTO *table\_2 (column\_1, column\_2, …)*

SELECT *column\_1, column\_2, …*

FROM *table\_name*

WHERE *condition*

* + 1. **Delete table**

DROP TEMPORARY TABLE name\_table;

1. **CTE** (Common Table Expression)

* Giúp đơn giản hóa trong việc viết những câu query có nhiều Join và Subquery
* CTE được coi là một table tạm có khả năng "self-referencing" và được phép sử dụng nhiều lần trong cùng 1 query.
* Được dùng để
  + Tạo Recursive Queries (đệ quy)
  + Thay thế View, Temporary trong 1 số trường hợp
  + Sử dụng kết quả của 1 CTE nhiều lần trong cùng 1 Query
  + Cho phép sử dụng lệnh Group By trên subquery có kết quả là duy nhất một column và 1 row
* Cú pháp:
  + **Basic**

WITH *CTE\_name* (*column\_name\_1, column\_name\_2*, …) AS (

*CTE\_query*

)

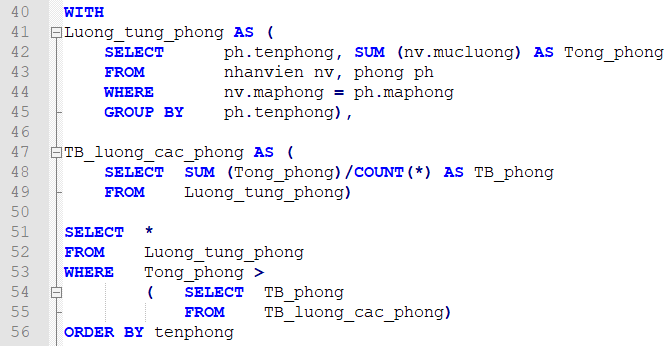
-- Statement using the CTE

SELECT | INSERT | DELETE | UPDATE | CREATE VIEW …

FROM *CTE\_name*;

WHERE *condition*

VD: Sử dụng mệnh đề WITH cho biết tên các phòng ban và tổng lương của các nhân viên trong phòng thỏa điều kiện là tổng lương của phòng phải lớn hơn trung bình tổng lương các phòng.



* + **Recursive Query**

WITH CTE\_name (*column\_name\_1, column\_name\_2*, …) AS (

--Anchor member is defined

*CTE\_query\_definition*.

UNION ALL

-- Recursive member is defined referencing cte\_name.

*CTE\_query\_definition*

)

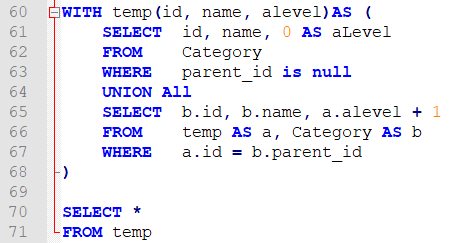
-- Statement using the CTE

SELECT | INSERT | DELETE | UPDATE | CREATE VIEW …

FROM *CTE\_name*

WHERE *Condition*

VD:



*Xem thêm ở đây:* [*https://khoaba.wordpress.com/2010/09/17/de-quy-voi-lenh-with-trong-sql-server/*](https://khoaba.wordpress.com/2010/09/17/de-quy-voi-lenh-with-trong-sql-server/)