BÀI THỰC HÀNH SỐ 2

I. Mục tiêu.

- Hiện thực được ngôn ngữ thao tác dữ liệu và truy vấn đơn giản SQL trên hệ quản trị SQL Server
- Các lệnh thao tác dữ liệu: thêm, xóa, sửa
- Cú pháp lệnh truy vấn (select ... from... where)
- Giới thiệu Select ... into …, view
- Truy vấn con (truy vấn lồng)
- Thực hiện các phép hội, giao, trừ

II. Hướng dẫn chung

1. Lệnh thêm dữ liệu (INSERT)

```
- INSERT INTO tên_bảng (cột1,…,cột n) VALUES (giá_trị_1,….,
giá_trị_n)
- INSERT INTO tên_bảng VALUES (giá_trị_1, giá_trị_2,…,
giá_trị_n)
- SELECT * INTO tên-bảng-mới from tên-bảng-có-sẵn
- INSERT INTO tên-bảng-tạo-trước select * from tên-bảng-
có-sẵn
```

2. Lệnh sửa dữ liệu (UPDATE)

```
UPDATE tên_bảng
SET cột_1 = giá_trị_1, cột_2 = giá_trị_2 …
[WHERE điều_kiện]
```

3. Lệnh xóa dữ liệu (DELETE)

```
DELETE FROM tên_bảng [WHERE điều_kiện]
```

4. Truy vấn dữ liệu

Câu truy vấn tổng quát

SELECT [DISTINCT] danh_sách_cột

FROM danh sách các quan hệ (hay bảng, table)

[WHERE điều_kiện]

[GROUP BY danh_sách_cột_gom_nhóm]

[HAVING điều_kiện_trên_nhóm]

[ORDER BY cột1 ASC | DESC, cột2 ASC | DESC,…]

Dạng truy vấn đơn giản

```
SELECT <Danh sách các cột> FROM <Danh sách Bảng>
SELECT * FROM <Tên Bảng>
```

Mệnh đề where: để xác định điều kiện lấy dữ liệu

```
SELECT <Danh sách các cột> FROM <Tên bảng> WHERE …
```

Ví dụ:

```
SELECT MaSV, HoDem, Ten FROM SINHVIEN WHERE GioiTinh = "Nữ";

SELECT MaSV, HoDem, Ten
FROM SINHVIEN
WHERE GioiTinh = "Nam" and MaLop = "CSDL-IT004";
```

5. View

View giúp tạo ra các bảng ảo (Virtual Table) chứa các thông tin, dữ liệu đã được lưu trữ sẵn cho người dùng sử dụng. View không tồn tại như một cấu trúc lưu trữ dữ liệu trong CSDL.

View thường được sử dụng với các mục đích sau:

- Tập trung (focus) trên những dữ liệu đã được xác định.
- Đơn giản hóa thao tác dữ liệu
- Trích xuất dữ liệu tạo báo cáo.
- Bảo mật dữ liệu, che giấu thông tin…

```
CREATE VIEW tên_view AS
SELECT cột_1, cột_2...
FROM tên_bảng
WHERE [điều kiện];
```

6. DISTINCT

Từ khóa **DISTINCT** trong SQL được sử dụng kết hợp với câu lệnh SELECT để loại bỏ tất cả các bộ trùng lặp và chỉ lấy các bộ duy nhất.

```
SELECT DISTINCT column1, column2,...columnN
FROM table_name
WHERE [condition]
```

7. BETWEEN

BETWEEN được dùng để lấy dữ liệu trong một khoảng, trong lệnh SELECT, INSERT, UPDATE hoặc DELETE.

```
SELECT (tên) cột1, cột2, ..
FROM tên_bảng
WHERE tên_cột BETWEEN giatri1 AND giatri2;
```

8. IS NULL, IS NOT NULL

Điều kiện IS NULL và IS NOT NULL trong SQL được sử dụng để chỉ ra một điều kiện rằng giá trị của một cột phải bằng hoặc khác null.

```
SELECT (tên) cột1, cột2, ..
FROM tên_bảng
WHERE tên_cột IS NOT NULL;
```

9. IN & NOT IN

Kiểm tra có nằm (hoặc không nằm) trong một dãy giá trị hay không.

```
SELECT columns_list
FROM table_name
WHERE column_name IN (list_value);
```

10. SOME, ALL, ANY

Toán tử SOME có thể dùng để so sánh một giá trị với từng giá trị trong tập dữ liệu. SOME trả về TRUE nếu có một giá trị bất kỳ trong tập dữ liệu này thỏa mãn điều kiện.

Toán tử ALL trong SQl có thể sử dụng để lấy toàn bộ dữ liệu của các cột của một bảng trong mệnh đề SELECT. Hoặc nó cũng có thể sử dụng để so sánh một giá trị với tất cả các giá trị trong một tập kết quả khác.

Toán tử ANY có thể dùng để so sánh một giá trị với từng giá trị trong tập dữ liệu. ANY trả về TRUE nếu có một giá trị bất kỳ trong tập dữ liệu này thỏa mãn điều kiện.

Toán tử SOME và ANY về cơ bản hoàn toàn giống nhau, chúng ta có thể sử dụng một trong hai tùy ý.

```
SELECT column_name(s)
FROM table_name
WHERE expression comparison_operator SOME (subquery)
```

```
SELECT ALL field_name
FROM table_name
WHERE condition(s);

SELECT column_name(s)
FROM table_name
WHERE column_name comparison_operator ALL
(SELECT column_name
FROM table_name
WHERE condition(s));

SELECT column_name(s)
FROM table_name
WHERE column_name comparison_operator ANY
(SELECT column_name
FROM table_name
WHERE condition(s));
```

11. LIKE

Mệnh đề LIKE trong SQL được sử dụng để so sánh một giá trị với các giá trị tương tự sử dụng toán tử ký tự đại diện (wildcard). Có hai ký tự đại diện được sử dụng kết hợp với toán tử LIKE.

- Phần trăm (%)
- Dấu gạch dưới (_)

Dấu phần trăm thể hiện không, một hoặc nhiều ký tự. Dấu gạch dưới đại diện cho một số hoặc một ký tự. Những ký hiệu này có thể được sử dụng trong sự kết hợp.

```
SELECT FROM table_name
WHERE column LIKE 'XXXX%'or SELECT FROM table_name
WHERE column LIKE '%XXXX%'orSELECT FROM table_name
WHERE column LIKE 'XXXX_'orSELECT FROM table_name
WHERE column LIKE '_XXXX'orSELECT FROM table_name
WHERE column LIKE '_XXXX_'
```

Ví dụ:

WHERE SALARY LIKE '20%' --Tìm trong cột lương bất kỳ giá trị nào bắt đầu bằng 20.

WHERE SALARY LIKE '%2' --Tìm trong cột lương bất kỳ giá trị nào kết thúc bằng 2.

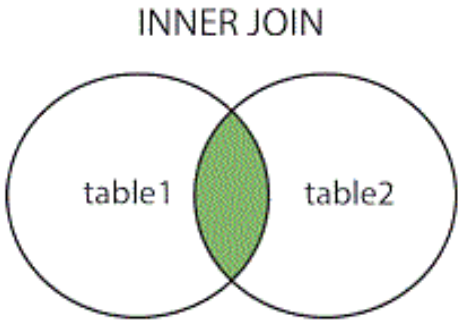
12. JOIN

Mệnh đề **JOIN** trong SQL được sử dụng để kết hợp các bản ghi từ hai hay nhiều bảng trong cơ sở dữ liệu. JOIN là một phương tiện để kết hợp các trường từ hai bảng bằng cách sử dụng các giá trị chung cho mỗi bảng.

```
SELECT A.column1, A.column2, B.column1, B.column3
FROM table1 A, table2 B
WHERE A.column1 = B.column2;
```

a. Inner Join

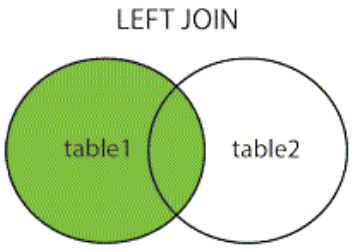
Lệnh **Inner Join** trong SQL tạo một bảng kết quả mới bằng cách kết hợp các giá trị cột của hai bảng (table1 và table2) dựa trên điều kiện nối. Truy vấn so sánh từng hàng của table1 với từng hàng của table2 để tìm tất cả các cặp hàng thỏa mãn điều kiện nối. Khi điều kiện nối được thỏa mãn, các giá trị cột cho mỗi cặp hàng A và B phù hợp sẽ được kết hợp thành một hàng kết quả.




```
SELECT table1.column1, table2.column2...
FROM table1
INNER JOIN table2
ON table1.common_field = table2.common_field;
```

b. Left Join

Lệnh **Left Join** trong SQL trả về tất cả các hàng từ bảng bên trái, ngay cả khi không có kết quả khớp nào trong bảng bên phải. Điều này có nghĩa là nếu mệnh đề ON không khớp với các bản ghi của bảng bên phải. Lệnh Left Join vẫn sẽ trả về một hàng trong kết quả, nhưng với NULL cho mỗi cột của bảng bên phải.

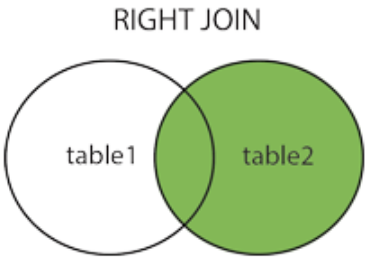



```
SELECT table1.column1, table2.column2...
FROM table1
LEFT JOIN table2
ON table1.common_field = table2.common_field;
```

c. Right Join

Lệnh Right Join trong SQL trả về tất cả các hàng từ bảng bên phải, ngay cả khi không có kết quả khớp nào trong bảng bên trái. Điều này có nghĩa là nếu mệnh đề ON không khớp

với các bản ghi của bảng bên trái. Lệnh Right Join vẫn sẽ trả về một hàng trong kết quả, nhưng với NULL cho mỗi cột của bảng bên trái.

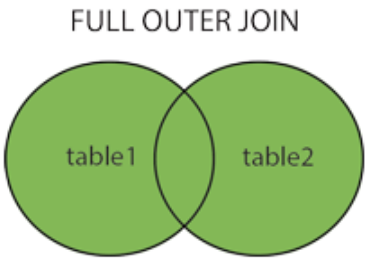



```
SELECT table1.column1, table2.column2...
FROM table1
RIGHT JOIN table2
ON table1.common_field = table2.common_field;
```

d. Full Join

Lệnh **Full Join** trong SQL là sự kết hợp các kết quả của cả hai phép nối Left Join và Right Join.

Kết quả join sẽ chứa tất cả các bản ghi từ cả hai bảng và điền vào NULL cho các kết quả bị thiếu ở hai bên.




```
SELECT table1.column1, table2.column2...
FROM table1
FULL JOIN table2
ON table1.common_field = table2.common_field;
```

e. Self Join

Lệnh **Self Join** trong SQL được sử dụng để nối một bảng với chính nó.

```
SELECT a.column_name, b.column_name...
FROM table1 a, table1 b
WHERE a.common_field = b.common_field;
```

13. UNION

Mệnh đề **UNION** trong SQL được sử dụng để kết hợp các kết quả của hai hoặc nhiều câu lệnh SELECT mà không cần trả về bất kỳ hàng trùng lặp nào.

```
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
UNION
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
```

UNION ALL được sử dụng để kết hợp các kết quả của hai câu lệnh SELECT bao gồm cả các hàng trùng lặp.

```
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
UNION ALL
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
```

14. INTERSECT

Toán tử **INTERSECT** được dùng để trả về các bản ghi nằm trong cả 2 bộ dữ liệu hoặc lệnh SELECT. Nếu một bản ghi chỉ có trong một truy vấn chứ không có trong truy vấn còn lại, nó sẽ bị loại bỏ khỏi bộ kết quả của INTERSECT.

```
SELECT cot1, cot2, .. cotN
FROM bang1
[WHERE dieu_kien1]
INTERSECT
SELECT cot1, cot2, .. cotN
FROM bang2
[WHERE dieu_kien2];
```

15. EXCEPT

Toán tử **EXCEPT** trong SQL Server được dùng để trả về các hàng trong lệnh SELECT đầu tiên mà không trả về trong lệnh SELECT thứ hai. Mỗi lệnh SELECT sẽ có một bộ dữ liệu. Toán tử EXCEPT lấy bản ghi từ bộ thứ 1 và bỏ các kết quả từ bộ 2.

```
SELECT cot1, cot2, .. cotN
FROM bang1
[WHERE dieu_kien1]
EXCEPT
SELECT cot1, cot2, .. cotN
FROM bang2
[WHERE dieu_kien2];
```

16. EXIST

Điều kiện **EXISTS** được dùng để kết hợp với truy vấn nội bộ (subquery). Điều kiện được đáp ứng nếu truy vấn nội bộ trả về ít nhất 1 hàng. Điều kiện này có thể dùng trong lệnh SELECT, INSERT, UPDATE hoặc DELETE.

```
WHERE EXISTS (subquery);
```

Cách dùng NOT với điều kiện EXIST trong SQL Server: Lệnh NOT được dùng để trích xuất những bản ghi điều kiện WHERE được cung cấp bởi người dùng là NOT TRUE hoặc FALSE

17. ORDER BY

Mệnh đề ORDER BY được sử dụng để sắp xếp dữ liệu theo thứ tự tăng dần hoặc theo thứ tự giảm dần trên một hoặc nhiều cột. Một số cơ sở dữ liệu mặc định sắp xếp các kết quả truy vấn theo thứ tự tăng dần.

Trong đó, lệnh ASC được sử dụng để sắp xếp tăng dần và DESC được sử dụng để sắp xếp giảm dần.

```
SELECT danhsach_cot
FROM ten_bang
[WHERE dieu_kien]
[ORDER BY cot1, cot2, .. cotN] [ASC | DESC];
```

III. Bài tập thực hành

Câu 1: Bài QuanLyBanHang:

I.8 --> I.10,

II.1. Nhập dữ liệu

II.2 -->II.4, II.5(optional)

III.1 -->III.17

Câu 2: Bài QuanLyGiaoVu

Hoàn thành từ câu 1 đến 8 (phần I). Câu 2 không làm.

Làm từ câu 1 và 3 trong phần II

Làm từ câu 1 đến 4 trong phần III

Quy định:

Bài tập làm theo cá nhân

Không nộp bài là 0 điểm hoặc các bài làm giống nhau sẽ chia cho số người giống nhau. Các bài đặt tên như sau:

- File bài tập về nhà đặt tên theo qui tắc MSSV_ Lab2_HW.sql, ở đầu file có ghi chú Họ tên và MSSV, từng câu có ghi chú số thứ tự từng câu và nội dung câu hỏi .
- Nếu bài nộp gồm nhiều file sql khác nhau thì nén thành file zip và đặt tên file zip giống như trên.