

## Bài thực hành tuần 4

### Truy vấn cơ bản (phần 2)

#### ❖ Nội dung chính

Trong bài này, sẽ đề cập đến cách sử dụng một số toán tử như *IN*, *BETWEEN*, *UNION*, *LIKE*, *ORDER BY*; Thuộc tính suy diễn.

#### 1. Toán tử IN

Toán tử IN cho phép chọn giá trị phù hợp từ một tập các giá trị. Cú pháp sử dụng như sau:

```
SELECT danh sách các cột  
FROM tên bảng  
WHERE cột IN ("giá trị 1", "giá trị 2"...) 
```

Các cột trong mệnh đề WHERE không cần phải xuất hiện trong danh sách cột đã chọn, nhưng nó phải là một cột trong bảng. Nếu danh sách có nhiều hơn một giá trị, mỗi mục được phân cách bằng dấu phẩy. Ngoài ra, có thể sử dụng toán tử NOT đi kèm với toán tử IN cho mục đích phủ định.

Chúng ta hãy xem một số ví dụ sau:

Giả sử nếu muốn tìm tất cả các văn phòng được đặt tại Mỹ (USA) và Pháp (France), có thể thực hiện truy vấn sau đây:

```
SELECT officeCode, city, phone  
FROM offices  
WHERE country = 'USA' OR country = 'France' 
```

Trong trường hợp này, chúng ta có thể sử dụng IN thay vì truy vấn trên:

```
SELECT officeCode, city, phone  
FROM offices  
WHERE country IN ('USA', 'France') 
```

Kết quả trả về như sau:

	officeCode	city	phone
▶	1	San Francisco	+1 650 219 4782
	2	Boston	+1 215 837 0825
	3	NYC	+1 212 555 3000
	4	Paris	+33 14 723 4404

Để có được tất cả các văn phòng không nằm ở Mỹ và Pháp, chúng ta có thể sử dụng NOT IN như sau:

```
SELECT officeCode, city, phone
FROM offices
WHERE country NOT IN ('USA', 'France')
```

Kết quả trả về như sau:

Kết quả trả về như sau:

	officeCode	city	phone
▶	5	Tokyo	+81 33 224 5000
	6	Sydney	+61 2 9264 2451
	7	London	+44 20 7877 2041

## 2. Toán tử BETWEEN

BETWEEN cho phép lấy các giá trị trong một phạm vi cụ thể. Nó phải được sử dụng trong mệnh đề WHERE. Sau đây minh họa cú pháp:

```
SELECT column_list
FROM table_name
WHERE column_1 BETWEEN lower_range AND upper_range
```

MySQL trả lại tất cả bản ghi trong đó giá trị column\_1 nằm trong phạm vi lower\_range và upper\_range. Truy vấn tương đương để có được cùng một kết quả là:

```
SELECT column_list
FROM table_name
WHERE column_1 >= lower_range AND column_1 <= upper_range
```

**Ví dụ:**

Giả sử chúng ta muốn tìm tất cả các sản phẩm có giá nằm trong phạm vi 90 \$ và 100 \$, chúng ta có thể thực hiện truy vấn sau đây:

```
SELECT productCode, ProductName, buyPrice
FROM products
WHERE buyPrice BETWEEN 90 AND 100
ORDER BY buyPrice DESC
```

Kết quả trả về như sau:

	productCode	ProductName	buyPrice
►	S10_1949	1952 Alpine Renault 1300	98.58
	S24_3856	1956 Porsche 356A Coupe	98.3
	S12_1108	2001 Ferrari Enzo	95.59
	S12_1099	1968 Ford Mustang	95.34
	S18_1984	1995 Honda Civic	93.89
	S18_4027	1970 Triumph Spitfire	91.92
	S10_4698	2003 Harley-Davidson Eagle Drag Bike	91.02

Để tìm tất cả các bản ghi không nằm trong một phạm vi, chúng ta sử dụng NOT BETWEEN. Ví dụ: để tìm tất cả các sản phẩm với giá mua nằm ngoài phạm vi 20 và 100, chúng ta có thể viết truy vấn sau đây:

```
SELECT productCode, ProductName, buyPrice
FROM products
WHERE buyPrice NOT BETWEEN 20 AND 100
```

Kết quả trả về như sau:

	productCode	ProductName	buyPrice
►	S10_4962	1962 LanciaA Delta 16V	103.42
	S18_2238	1998 Chrysler Plymouth Prowler	101.51
	S24_2840	1958 Chevy Corvette Limited Edition	15.91
	S24_2972	1982 Lamborghini Diablo	16.24

Truy vấn trên tương đương với truy vấn sau:

```
SELECT productCode, ProductName, buyPrice
FROM products
WHERE buyPrice < 20 OR buyPrice > 100
ORDER BY buyPrice DESC
```

Kết quả trả về như sau:

	productCode	ProductName	buyPrice
▶	S10_4962	1962 LanciaA Delta 16V	103.42
	S18_2238	1998 Chrysler Plymouth Prowler	101.51
	S24_2972	1982 Lamborghini Diablo	16.24
	S24_2840	1958 Chevy Corvette Limited Edition	15.91

### 3. Toán tử LIKE

LIKE cho phép thực hiện việc tìm kiếm thông tin dựa trên sự so sánh ký tự ('giống như'). LIKE thường được sử dụng với câu lệnh SELECT trong mệnh đề WHERE. MySQL cung cấp cho hai ký tự đại diện sử dụng với LIKE, đó là % và \_.

- Ký tự đại diện tỷ lệ phần trăm (%) đại diện cho bất kỳ chuỗi có thể không có hoặc có nhiều ký tự
- Gạch dưới (\_) chỉ đại diện cho một ký tự duy nhất.

**Ví dụ:** Giả sử muốn tìm kiếm những nhân viên có tên bắt đầu với ký tự 'a', có thể làm điều đó như sau:

```
SELECT employeeNumber, lastName, firstName
FROM employees
WHERE firstName LIKE 'a%'
```

Kết quả trả về như sau:

	employeeNumber	lastName	firstName
▶	1143	Bow	Anthony
	1611	Fixter	Andy

MySQL quét toàn bộ bảng employees (*nhân viên*) để tìm tất cả nhân viên có tên bắt đầu với ký tự 'a' và theo sau bởi một số lượng ký tự bất kỳ.

**Ví dụ:** Để tìm kiếm tất cả các nhân viên có họ kết thúc với chuỗi 'on', có thể thực hiện truy vấn như sau:

```
SELECT employeeNumber, lastName, firstName
FROM employees
WHERE lastName LIKE '%on'
```

Kết quả trả về như sau:

	employeeNumber	lastName	firstName
►	1056	Patterson	Mary
	1088	Patterson	William
	1166	Thompson	Leslie
	1216	Patterson	Steve

Nếu chỉ biết rằng chuỗi tìm kiếm được nhúng vào một vị trí nào đó trong giá trị của một cột, có thể đặt % đầu và cuối của chuỗi tìm kiếm để tìm tất cả khả năng.

**Ví dụ:** muốn tìm tất cả các nhân viên mà họ của các nhân viên này có chứa cụm 'on', có thể thực hiện truy vấn sau đây:

```
SELECT employeeNumber, lastName, firstName
FROM employees
WHERE lastName LIKE '%on%'
```

Kết quả trả về như sau:

	employeeNumber	lastName	firstName
►	1056	Patterson	Mary
	1088	Patterson	William
	1102	Bondur	Gerard
	1166	Thompson	Leslie
	1216	Patterson	Steve
	1337	Bondur	Loui
	1504	Jones	Bary

Chúng ta cũng có thể dùng NOT kèm với LIKE để hàm chứa ý nghĩa phủ định.

**Ví dụ:** muốn tìm các nhân viên có họ không bắt đầu bởi ký tự 'B', viết như sau:

```
SELECT employeeNumber, lastName, firstName
FROM employees
WHERE lastName NOT LIKE 'B%'
```

Kết quả trả về như sau:

	employeeNumber	lastName	firstName
▶	1002	Murphy	Diane
	1056	Patterson	Mary
	1076	Firrelli	Jeff
	1088	Patterson	William
	1165	Jennings	Leslie
	1166	Thompson	Leslie
	1188	Firrelli	Julie
	1216	Patterson	Steve
	1286	Tseng	Foon Yue
	1323	Vanauf	George
	1370	Hernandez	Gerard
	1401	Castillo	Pamela
	1504	Jones	Barry
	1611	Fixter	Andy
	1612	Marsh	Peter

Lưu ý là MySQL không phân biệt chữ hoa chữ thường nên 'b%' và 'B%' là như nhau.

Trong trường hợp chuỗi tìm kiếm của lại bắt đầu bởi một ký tự đại diện, chẳng hạn là '\_', mysql cung cấp cho ký tự '\' để chỉ ra rằng các ký tự đại diện đi sau đó được sử dụng theo đúng nghĩa đen chứ không còn là ký tự đại diện nữa.

Ví dụ: tìm các sản phẩm mà mã của chúng có chứa chuỗi '\_20', khi đó phải viết truy vấn như sau:

```
SELECT productCode, productName
FROM products
```

```
WHERE productCode LIKE '%\_20%'
```

Kết quả trả về như sau:

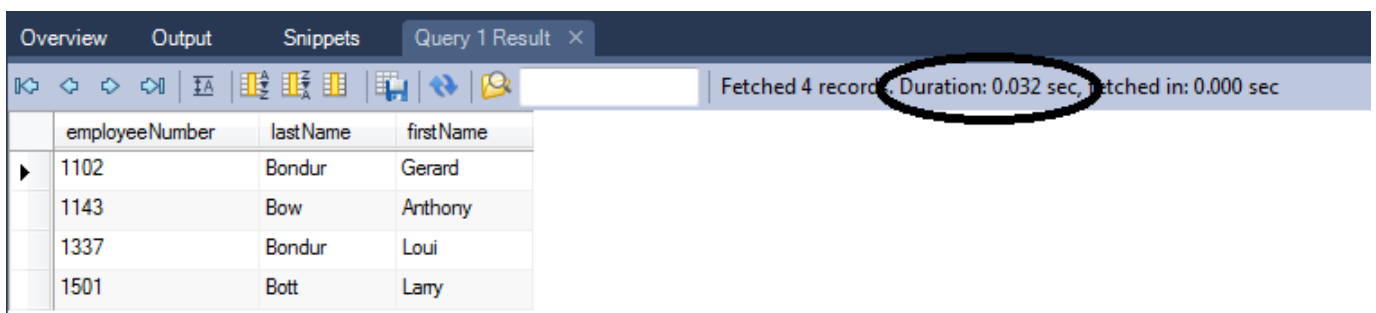
	productCode	productName
►	S10_2016	1996 Moto Guzzi 1100i
	S24_2000	1960 BSA Gold Star DBD34
	S24_2011	18th century schooner
	S24_2022	1938 Cadillac V-16 Presidential Limousine
	S700_2047	HMS Bounty

LIKE cung cấp cho một cách thuận tiện để tìm bản ghi có các cột chứa các chuỗi phù hợp với mẫu tìm kiếm. Tuy nhiên, do việc thực thi LIKE chính là quét toàn bộ bảng để tìm tất cả các bản ghi phù hợp do đó nó không cho phép database engine sử dụng index để tìm kiếm nhanh. Khi dữ liệu trong bảng là đủ lớn, hiệu suất thực thi LIKE sẽ bị suy giảm. Trong một số trường hợp, có thể tránh vấn đề này bằng cách sử dụng các kỹ thuật khác để đạt được các kết quả tương tự. Hoặc sử dụng phương pháp đánh chỉ mục FULLTEXT của MySQL (sẽ đề cập về kỹ thuật này trong khóa học về Hệ quản trị CSDL MySQL).

**Ví dụ:** nếu muốn tìm tất cả các nhân viên có tên đầu tiên bắt đầu với một chuỗi quy định có thể sử dụng hàm LEFT() giống như các truy vấn sau đây:

```
SET @str = 'b';  
SELECT employeeNumber, lastName, firstName  
FROM employees  
WHERE LEFT(lastname, length(@str)) = @str;
```

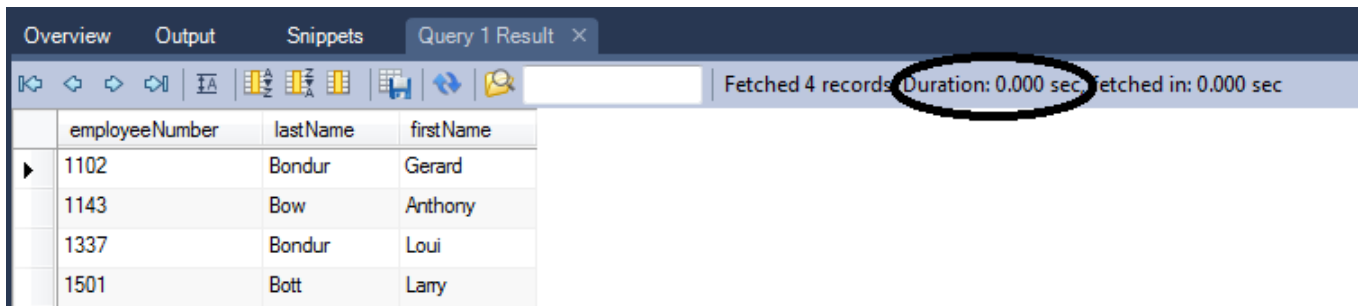
Kết quả trả về như sau:



	employeeNumber	lastName	firstName
►	1102	Bondur	Gerard
	1143	Bow	Anthony
	1337	Bondur	Loui
	1501	Bott	Larry

Kết quả trả về của truy vấn này là tương đương với truy vấn dưới đây, tuy nhiên tốc độ thực thi của cách viết sau tốt hơn rất nhiều vì chúng ta có thể sử dụng index trên cột *lastname*.

```
SELECT employeeNumber, lastName, firstName
FROM employees
WHERE lastname LIKE 'b%'
```



The screenshot shows a database query result interface. At the top, there are tabs for 'Overview', 'Output', 'Snippets', and 'Query 1 Result'. Below the tabs is a toolbar with various icons. To the right of the toolbar, it says 'Fetched 4 records', 'Duration: 0.000 sec', and 'Fetched in: 0.000 sec'. The main area displays a table with the following data:

	employeeNumber	lastName	firstName
▶	1102	Bondur	Gerard
	1143	Bow	Anthony
	1337	Bondur	Loui
	1501	Bott	Larry

#### 4. Thuộc tính suy diễn (Derived Attribute)

SQL cung cấp khả năng tạo các thuộc tính suy diễn trong bảng kết quả trả về sử dụng các toán tử và hàm dựa trên các thuộc tính có sẵn. Tên cột của thuộc tính suy diễn phụ thuộc vào hệ thống, tuy nhiên có thể gán bí danh làm tên cột.

Ví dụ sau sẽ tạo ra một cột suy diễn được đặt tên là *lineTotal*, thuộc tính này là kết quả phép nhân giữa hai thuộc tính *priceEach* và *quantityOrdered*

```
SELECT orderNumber, (priceEach*quantityOrdered) as
lineTotal
FROM orderdetails
```



	orderNumber	lineTotal
▶	10100	4080
	10100	2754.5
	10100	1660.12
	10100	1729.21
	10101	2701.5
	10101	4343.56
	10101	1463.8500000000001
	10101	2040.1000000000001
	10102	3726.45
	10102	1768.3300000000002
	10103	5571.8
	10103	5026.14
	10103	3284.28
	10103	3307.5
	10103	1283.48
	10103	2400.12

## 5. Sắp xếp kết quả với ORDER BY

Mệnh đề ORDER BY cho phép sắp xếp các kết quả trên một hoặc nhiều cột trong kết quả truy vấn theo thứ tự tăng dần hay giảm dần. Để sắp xếp kết quả theo thứ tự tăng dần, sử dụng ASC; giảm dần là DESC. Theo mặc định, ORDER BY sẽ sắp xếp các kết quả theo thứ tự tăng dần.

**Ví dụ:** để sắp xếp danh sách nhân viên theo *tên* và *vị trí công việc*, có thể thực hiện truy vấn sau đây:

```
SELECT FirstName, LastName, jobtitle
FROM Employees
ORDER BY firstname ASC, jobtitle DESC;
```

	FirstName	LastName	jobtitle
►	Andy	Fixter	Sales Rep
	Anthony	Bow	Sales Manager (NA)
	Barry	Jones	Sales Rep
	Diane	Murphy	President
	Foon Yue	Tseng	Sales Rep
	George	Vanauf	Sales Rep
	Gerard	Hernandez	Sales Rep
	Gerard	Bondur	Sale Manager (EMEA)
	Jeff	Firelli	VP Marketing
	Julie	Firelli	Sales Rep
	Larry	Bott	Sales Rep
	Leslie	Jennings	Sales Rep
	Leslie	Thompson	Sales Rep
	Loui	Bondur	Sales Rep
	Mami	Nishi	Sales Rep

Hoặc có thể đưa ra thông tin về tên các sản phẩm theo thứ tự tăng dần của số lượng hàng tồn kho bằng truy vấn như sau:

```
SELECT productName
FROM Products
ORDER BY quantityInStock;
```

Trong câu lệnh trên từ khóa ASC không sử dụng, do mặc định sẽ sắp xếp kết quả theo thứ tự tăng dần. Kết quả của câu lệnh trong hình sau.

	productName
▶	1960 BSA Gold Star DBD34
	1968 Ford Mustang
	1928 Ford Phaeton Deluxe
	1997 BMW F650 ST
	Pont Yacht
	1911 Ford Town Car
	1928 Mercedes-Benz SSK
	F/A 18 Homet 1/72
	2002 Yamaha YZR M1
	The Mayflower
	1996 Peterbilt 379 Stake Bed with Outtrigger
	P-51-D Mustang
	1970 Chevy Chevelle SS 454
	Diamond T620 Semi-Skirted Tanker
	1969 Ford Falcon

Nếu không chỉ rõ việc sắp xếp được thực hiện theo thứ tự tăng hay giảm dần, MySQL sẽ mặc định việc sắp xếp dữ liệu được thực hiện theo thứ tự tăng dần.

## 6. Kết hợp các kết quả với toán tử UNION

UNION cho phép kết hợp hai hoặc nhiều bộ kết quả từ nhiều bảng với nhau. Cú pháp của việc sử dụng MySQL UNION là như sau:

```
SELECT statement
UNION [DISTINCT | ALL]
SELECT statement
UNION [DISTINCT | ALL]
...
```

Để sử dụng UNION, có một số nguyên tắc cần phải làm theo:

- Số lượng các cột trong mỗi câu lệnh SELECT phải giống nhau.
- Các kiểu dữ liệu của cột trong danh sách cột của câu lệnh SELECT phải giống nhau hoặc ít nhất là có thể chuyển đổi sang cho nhau.

Theo mặc định, UNION MySQL loại bỏ tất cả các hàng trùng lặp từ kết quả ngay cả khi không sử dụng từ khoá DISTINCT sau từ khoá UNION.

Nếu sử dụng UNION ALL, các hàng trùng lặp vẫn còn trong tập hợp kết quả cuối cùng. chỉ nên sử dụng điều này trong các trường hợp hoặc là muốn giữ lại bản sao các hàng, hoặc chắc chắn rằng có không có bản sao các hàng trong tập hợp kết quả.

**Ví dụ:** kết hợp thông tin về các khách hàng và nhân viên thành một tập hợp kết quả, sử dụng truy vấn sau đây:

```
SELECT customerNumber id, contactLastname name
FROM customers
UNION
SELECT employeeNumber id,firstname name
FROM employees
```

	id	name
▶	103	Schmitt
	112	King
	114	Ferguson
	119	Labruno
	121	Bergulfsen
	124	Nelson
	125	Piestrzeniewicz
	128	Keitel
	129	Murphy
	131	Lee
	141	Freyre
	144	Berglund
	145	Petersen
	146	Saveley
	148	Natividad
	151	Young

Khi sử dụng ORDER BY để sắp xếp kết quả với UNION, phải đặt nó ở vị trí cuối cùng trong mệnh đề SELECT.

**Ví dụ:** Giả sử kết hợp thông tin của nhân viên và khách hàng, sau đó muốn sắp xếp kết quả theo *tên* và *ID* thứ tự tăng dần

```
(SELECT customerNumber, contactLastname
FROM customers)
UNION
(SELECT employeeNumber, firstname
FROM employees)
ORDER BY contactLastname, customerNumber
```

	customerNumber	contactLastname
►	249	Accorti
	481	Altagar,G M
	307	Andersen
	1611	Andy
	1143	Anthony
	465	Anton
	187	Ashworth
	204	Barajas
	1504	Bary
	462	Benitez
	240	Bennett
	144	Berglund
	121	Bergulfsen
	172	Bertrand
	321	Brown
	324	Brown

Nếu tên cột không giống nhau trong hai mệnh đề SELECT của phép UNION, tên nào sẽ được hiển thị ở đầu ra nếu chúng ta không sử dụng bí danh cho mỗi cột trong mệnh đề SELECT. Câu trả lời là MySQL sẽ sử dụng các tên cột của câu lệnh SELECT đầu tiên là tên cột trong kết quả đầu ra

```
(SELECT customerNumber, contactLastname
FROM customers)
UNION
```

```
(SELECT employeeNumber, firstname  
FROM employees)  
ORDER BY contactLastname, customerNumber
```

Kết quả của phép toán hợp giữa hai tập kết quả từ bảng dữ liệu customers và employees

	customerNumber	contactLastname
►	249	Accorti
	481	Altagar,G M
	307	Andersen
	1611	Andy
	1143	Anthony
	465	Anton
	187	Ashworth
	204	Barajas
	1504	Bary
	462	Benitez
	240	Bennett

MySQL cũng cung cấp một lựa chọn khác để sắp xếp các kết quả thiết lập dựa trên vị trí cột trong mệnh đề ORDER BY như truy vấn sau đây:

```
(SELECT customerNumber, contactLastname  
FROM customers)  
UNION  
(SELECT employeeNumber,firstname  
FROM employees)  
ORDER BY 2, 1
```

customerNumber	contactLastname
249	Accorti
481	Altagar,G M
307	Andersen
1611	Andy
1143	Anthony
465	Anton
187	Ashworth
204	Barajas
1504	Bamy
462	Benitez
240	Bennett
144	Berglund
121	Bergulfsen
172	Bertrand

#### ❖ Bài tập thực hành:

1. Dùng toán tử IN để đưa ra thông tin của các khách hàng sống tại các thành phố Nantes và Lyon. Viết cách khác sử dụng toán tử OR
2. Sử dụng BETWEEN để tìm các đơn hàng đã được chuyển trong khoảng thời gian từ '10/1/2003' đến '10/3/2003'. Viết cách khác sử dụng toán tử AND
3. Sử dụng LIKE để đưa ra thông tin về các nhóm hàng hoá có chứa từ 'CARS'.
4. Truy vấn 10 sản phẩm có số lượng trong kho là lớn nhất.
5. Đưa ra danh sách các sản phẩm và thêm thuộc tính là tiền hàng tồn của sản phẩm.