Bài thực hành số 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.*

# 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'

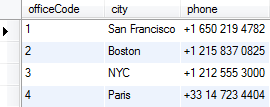
Trong trường hợp này, chúng ta có thể sử dụng IN thay vì truy vấn trên:

SELECT officeCode, city, phone

FROM offices

WHERE country IN ('USA','France')

Kết quả trả về như sau:



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

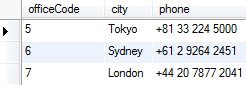
SELECT officeCode, city, phone

FROM offices

WHERE country NOT IN ('USA','France')

Kết quả trả về như sau:

Kết quả trả về như sau:



# 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\_rage 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:

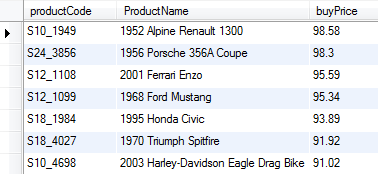
SELECT productCode,ProductName,buyPrice

FROM products

WHERE buyPrice BETWEEN 90 AND 100

ORDER BY buyPrice DESC

Kết quả trả về như sau:



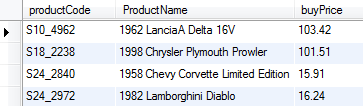
Để tìm tất cả các bản ghi không nằm trong một phạm vi, chúng ta sử dụng NOT BETWEEN. Ví dụ: để tìm tất cả các sản phẩm với giá mua nằm ngoài phạm vi 20 và 100, chúng ta có thể viết truy vấn sau đây:

SELECT productCode,ProductName,buyPrice

FROM products

WHERE buyPrice NOT BETWEEN 20 AND 100

Kết quả trả về như sau:



Truy vấn trên tương đương với truy vấn sau:

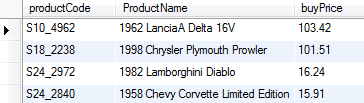
SELECT productCode,ProductName,buyPrice

FROM products

WHERE buyPrice < 20 OR buyPrice > 100

ORDER BY buyPrice DESC

Kết quả trả về như sau:



# 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:



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.

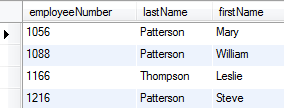
**Ví dụ:** Để tìm kiếm tất cả các nhân viên có họ kết thúc với chuỗi ‘on’, có thể thực hiện truy vấn như sau:

SELECT employeeNumber, lastName, firstName

FROM employees

WHERE lastName LIKE '%on'

Kết quả trả về như sau:



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.

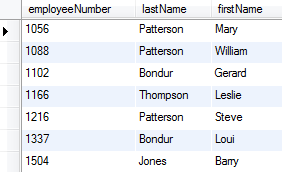
**Ví dụ:** muốn tìm tất cả các nhân viên mà họ của các nhân viên này có chứa cụm 'on', có thể thực hiện truy vấn sau đây:

SELECT employeeNumber, lastName, firstName

FROM employees

WHERE lastName LIKE '%on%'

Kết quả trả về như sau:



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

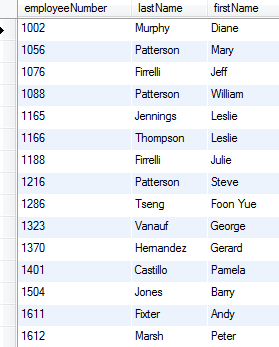
**Ví dụ:** muốn tìm các nhân viên có họ không bắt đầu bởi ký tự ‘B’, viết như sau:

SELECT employeeNumber, lastName, firstName

FROM employees

WHERE lastName NOT LIKE 'B%'

Kết quả trả về như sau:



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.

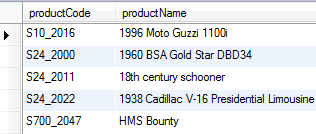
Ví dụ: tìm các sản phẩm mà mã của chúng có chứa chuỗi ‘\_20’, khi đó phải viết truy vấn như sau:

SELECT productCode, productName

FROM products

WHERE productCode LIKE '%\\_20%'

Kết quả trả về như sau:



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:

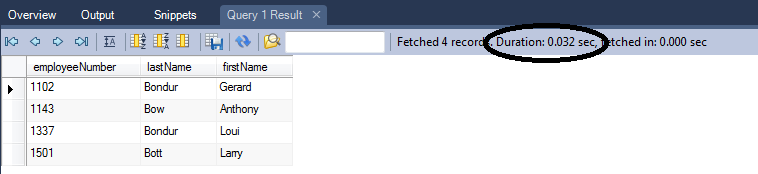
SET @str = 'b';

SELECT employeeNumber, lastName, firstName

FROM employees

WHERE LEFT(lastname,length(@str)) = @str;

Kết quả trả về như sau:

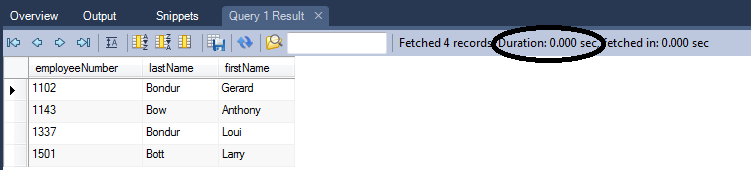


Kết quả trả về của truy vấn này là tương đương với truy vấn dưới đây, tuy nhiên tốc độ thực thi của cách viết sau tốt hơn rất nhiều vì chúng ta có thể sử dụng index trên cột *lastname*.

SELECT employeeNumber, lastName, firstName

FROM employees

WHERE lastname LIKE 'b%'



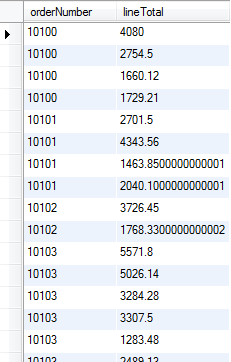
# Thuộc tính suy diễn (Derived Attribute)

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

****

# 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;

|  |
| --- |
|  |

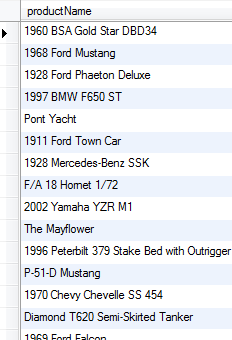
Hoặc có thể đưa ra thông tin về tên các sản phẩm theo thứ tự tăng dần của số lượng hàng tồn kho bằng truy vấn như sau:

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.



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.

# 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 cáckhách hàng và nhân viên thành một tập hợp kết quả, sử dụng truy vấn sau đây:

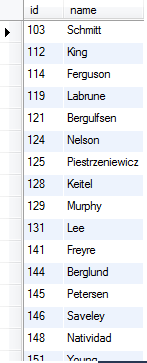
SELECT customerNumber id, contactLastname name

FROM customers

UNION

SELECT employeeNumber id,firstname name

FROM employees



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

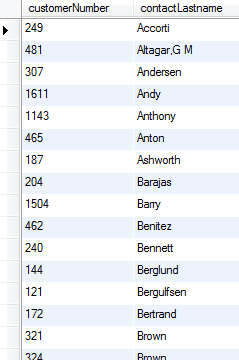
FROM customers)

UNION

(SELECT employeeNumber, firstname

FROM employees)

ORDER BY contactLastname, customerNumber



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)

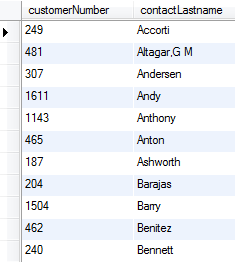
UNION

(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



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

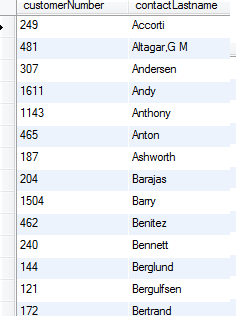
FROM customers)

UNION

(SELECT employeeNumber,firstname

FROM employees)

ORDER BY 2, 1



# 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.
2. Sử dụng BETWEEN để tìm các đơn hàng đã được chuyển trong khoảng thời gian từ ‘10/1/2003’ đến ‘10/3/2003’.
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.