**SQL CSDL**

SQL có thể làm gì?

* SQL có thể thực thi các truy vấn đối với cơ sở dữ liệu
* SQL có thể lấy dữ liệu từ cơ sở dữ liệu
* SQL có thể chèn các bản ghi trong cơ sở dữ liệu
* SQL có thể cập nhật các bản ghi trong cơ sở dữ liệu
* SQL có thể xóa các bản ghi từ cơ sở dữ liệu
* SQL có thể tạo cơ sở dữ liệu mới
* SQL có thể tạo các bảng mới trong cơ sở dữ liệu
* SQL có thể tạo các thủ tục được lưu trữ trong cơ sở dữ liệu
* SQL có thể tạo các khung nhìn trong cơ sở dữ liệu
* SQL có thể đặt quyền trên bảng, thủ tục và khung nhìn

Some of The Most Important SQL Commands

* **SELECT** - extracts data from a database
* **UPDATE** - updates data in a database
* **DELETE** - deletes data from a database
* **INSERT INTO** - inserts new data into a database
* **CREATE DATABASE** - creates a new database
* **ALTER DATABASE** - modifies a database
* **CREATE TABLE** - creates a new table
* **ALTER TABLE** - modifies a table
* **DROP TABLE** - deletes a table
* **CREATE INDEX** - creates an index (search key)
* **DROP INDEX** - deletes an index

## Câu lệnh SQL SELECT

Câu lệnh SELECT được sử dụng để chọn dữ liệu từ cơ sở dữ liệu.

Dữ liệu trả về được lưu trữ trong một bảng kết quả, được gọi là tập kết quả.

### **CHỌN Cú pháp**

SELECT column1, column2, ...  
FROM table\_name;

## Câu lệnh SQL CHỌN DISTINCT

Câu lệnh CHỌN DISTINCT được sử dụng để chỉ trả về các giá trị riêng biệt (khác nhau).

Trong một bảng, một cột thường chứa nhiều giá trị trùng lặp; và đôi khi bạn chỉ muốn liệt kê các giá trị (khác biệt) khác nhau.

**Khi dùng DISTINCT thì chỉ chả về 1 giá trị nếu như trong bảng có nhiều giá trị giống nhau**

### **CHỌN DISTINCT Cú pháp**

SELECT DISTINCT column1, column2, ...  
FROM table\_name;

## Bí danh SQL

Các bí danh SQL được sử dụng để đặt một bảng hoặc một cột trong bảng, một tên tạm thời.

Bí danh thường được sử dụng để làm cho tên cột dễ đọc hơn.

Một bí danh chỉ tồn tại trong suốt thời gian của truy vấn.

## SQL Aliases

**AS trong SQL**

SELECT Count(\*) AS **Demsoluong**  
FROM  Customers;

Ngoài ra ta còn có thể sử dụng từ khóa AS để khi muốn hiển thị rõ ràng một chỉ mục hoặc một giá trị.

**Lưu ý:** Nó yêu cầu dấu ngoặc kép hoặc dấu ngoặc vuông nếu tên bí danh chứa dấu cách:

SELECT CustomerName AS Customer, ContactName AS [Contact Person]  
FROM Customers;

Câu lệnh SQL sau đây tạo ra một bí danh có tên "Địa chỉ" kết hợp bốn cột (Địa chỉ, PostalCode, Thành phố và Quốc gia):

### **Thí dụ**

SELECT CustomerName, Address + ', ' + PostalCode + ' ' + City + ', ' + Country AS Address  
FROM Customers;

**Lưu ý:** Để câu lệnh SQL ở trên hoạt động trong MySQL, hãy sử dụng như sau:

SELECT CustomerName, CONCAT(Address,', ',PostalCode,', ',City,', ',Country) AS Address  
FROM Customers;

**Demsoluong**

**12**

## Mệnh đề WHERE của SQL

Mệnh đề WHERE được sử dụng để lọc các bản ghi.

Mệnh đề WHERE được sử dụng để chỉ trích xuất những bản ghi đáp ứng một điều kiện cụ thể.

SELECT column1, column2, ...  
FROM table\_name  
WHERE condition;

Các toán tử SQL VÀ, HOẶC và KHÔNG

Mệnh đề WHERE có thể được kết hợp với các toán tử AND, OR và NOT.

Các toán tử AND và OR được sử dụng để lọc các bản ghi dựa trên nhiều điều kiện:

* Toán tử AND hiển thị một bản ghi nếu tất cả các điều kiện được phân tách bằng AND là TRUE.
* Toán tử OR hiển thị một bản ghi nếu bất kỳ điều kiện nào được phân tách bằng OR là TRUE.

Toán tử NOT hiển thị một bản ghi nếu (các) điều kiện KHÔNG ĐÚNG.

## Sắp xếp SQL theo từ khóa ORDER BY

Từ khóa ORDER BY được sử dụng để sắp xếp tập kết quả theo thứ tự tăng dần hoặc giảm dần.

Từ khóa ORDER BY sắp xếp các bản ghi theo thứ tự tăng dần theo mặc định. Để sắp xếp các bản ghi theo thứ tự giảm dần, hãy sử dụng từ khóa DESC.

SELECT column1, column2, ...  
FROM table\_name  
ORDER BY column1, column2, ... ASC|DESC;

## Câu lệnh SQL INSERT INTO

Câu lệnh INSERT INTO được sử dụng để chèn các bản ghi mới vào một bảng.

### **XÁC NHẬN VÀO Cú pháp**

Có thể viết câu lệnh INSERT INTO theo hai cách.

Cách đầu tiên chỉ định cả tên cột và giá trị được chèn:

INSERT INTO table\_name (column1, column2, column3, ...)  
VALUES (value1, value2, value3, ...);

Nếu bạn đang thêm giá trị cho tất cả các cột của bảng, bạn không cần chỉ định tên cột trong truy vấn SQL. Tuy nhiên, hãy đảm bảo thứ tự của các giá trị theo cùng thứ tự với các cột trong bảng. Cú pháp INSERT INTO sẽ như sau:

INSERT INTO table\_name  
VALUES (value1, value2, value3, ...);

## Giá trị NULL là gì?

Trường có giá trị NULL là trường không có giá trị.

Nếu một trường trong bảng là tùy chọn, có thể chèn bản ghi mới hoặc cập nhật bản ghi mà không cần thêm giá trị vào trường này. Sau đó, trường sẽ được lưu với giá trị NULL.

## Làm thế nào để kiểm tra giá trị NULL?

Không thể kiểm tra các giá trị NULL bằng các toán tử so sánh, chẳng hạn như =, <hoặc <>.

Thay vào đó, chúng ta sẽ phải sử dụng các toán tử IS NULL và IS NOT NULL.

### **Cú pháp của NULL**

SELECT column\_namesFROM table\_name  
WHERE column\_name IS NULL;

## Tuyên bố CẬP NHẬT SQL

Câu lệnh CẬP NHẬT được sử dụng để sửa đổi các bản ghi hiện có trong một bảng.

### **CẬP NHẬT Cú pháp**

UPDATE table\_name  
SET column1 = value1, column2 = value2, ...  
WHERE condition;

**Lưu ý:** Cẩn thận khi cập nhật hồ sơ trong bảng! Lưu ý mệnh đề WHERE trong câu lệnh CẬP NHẬT. Mệnh đề WHERE chỉ định (các) bản ghi nào sẽ được cập nhật. Nếu bạn bỏ qua mệnh đề WHERE, tất cả các bản ghi trong bảng sẽ được cập nhật!

## Câu lệnh XÓA SQL

Câu lệnh DELETE được sử dụng để xóa các bản ghi hiện có trong một bảng.

### **XÓA Cú pháp**

DELETE FROM table\_name WHERE condition;

**Lưu ý:** Cẩn thận khi xóa các bản ghi trong bảng! Lưu ý mệnh đề WHERE trong câu lệnh XÓA. Mệnh đề WHERE chỉ định (các) bản ghi nào sẽ bị xóa. Nếu bạn bỏ qua mệnh đề WHERE, tất cả các bản ghi trong bảng sẽ bị xóa!

# Điều khoản SQL TOP, LIMIT hoặc ROWNUM

Câu lệnh TOP(SQL,MS ACCESS), LIMIT(MySQL), ROWNUM(Oracle) dùng để giới hạn số bản ghi muốn lấy

**Cú pháp truy cập SQL Server / MS:**

SELECT TOP *number*|*percent* *column\_name(s)*  
FROM *table\_name*WHERE *condition*;

**Cú pháp của MySQL:**

SELECT *column\_name(s)*  
FROM *table\_name*WHERE *condition*  
LIMIT *number*;

**Cú pháp của Oracle:**

SELECT *column\_name(s)*  
FROM *table\_name*  
WHERE ROWNUM <= *number*;

## Các hàm SQL MIN () và MAX ()

Hàm MIN () trả về giá trị nhỏ nhất của cột đã chọn.

Hàm MAX () trả về giá trị lớn nhất của cột đã chọn.

### **Cú pháp MIN ()**

SELECT MIN(column\_name)  
FROM table\_name  
WHERE condition;

### **Cú pháp MAX ()**

SELECT MAX(column\_name)  
FROM table\_name  
WHERE condition;

## Các hàm SQL COUNT (), AVG () và SUM ()

Hàm COUNT () trả về số lượng hàng khớp với một tiêu chí đã chỉ định.

Hàm AVG () trả về giá trị trung bình của một cột số.

Hàm SUM () trả về tổng của một cột số.

### **Cú pháp () Cú pháp**

SELECT COUNT(column\_name)  
FROM table\_name  
WHERE condition;

### **Cú pháp AVG ()**

SELECT AVG(column\_name)  
FROM table\_name  
WHERE condition;

### **Cú pháp () Cú pháp**

SELECT SUM(column\_name)  
FROM table\_name  
WHERE condition;

## The SQL LIKE Operator

Toán tử LIKE được sử dụng trong mệnh đề WHERE để tìm kiếm một mẫu đã chỉ định trong một cột.

Có hai ký tự đại diện thường được sử dụng cùng với toán tử THÍCH:

* % - Dấu phần trăm đại diện cho không, một hoặc nhiều ký tự
* \_ - Dấu gạch dưới đại diện cho một ký tự

**Lưu ý:** MS Access sử dụng dấu hoa thị (\*) thay vì dấu phần trăm (%) và dấu hỏi (?) Thay vì dấu gạch dưới (\_).

Dấu phần trăm và dấu gạch dưới cũng có thể được sử dụng trong các kết hợp!

SELECT \* FROM Customers  
WHERE CustomerName LIKE 'a%';

SELECT \*

FROM Customers

WHERE CustomerName LIKE '%a' AND CustomerName LIKE '\_e%' ;

### **Ký tự đại diện trong MS Access**

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Description** | **Example** |
| \* | Represents zero or more characters | bl\* finds bl, black, blue, and blob |
| ? | Represents a single character | h?t finds hot, hat, and hit |
| [] | Represents any single character within the brackets | h[oa]t finds hot and hat, but not hit |
| ! | Represents any character not in the brackets | h[!oa]t finds hit, but not hot and hat |
| - | Represents a range of characters | c[a-b]t finds cat and cbt |
| # | Represents any single numeric character | 2#5 finds 205, 215, 225, 235, 245, 255, 265, 275, 285, and 295 |

### **Ký tự đại diện trong SQL Server**

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Description** | **Example** |
| % | Represents zero or more characters | bl% finds bl, black, blue, and blob |
| \_ | Represents a single character | h\_t finds hot, hat, and hit |
| [] | Represents any single character within the brackets | h[oa]t finds hot and hat, but not hit |
| ^ | Represents any character not in the brackets | h[^oa]t finds hit, but not hot and hat |
| - | Represents a range of characters | c[a-b]t finds cat and cbt |

## Toán tử SQL IN

Toán tử IN cho phép bạn chỉ định nhiều giá trị trong mệnh đề WHERE.

Toán tử IN là một tốc ký cho nhiều điều kiện OR.

### **IN Cú pháp**

SELECT column\_name(s)  
FROM table\_name  
WHERE column\_name IN (value1, value2, ...);

hoặc là:

SELECT *column\_name(s)*  
FROM *table\_name*  
WHERE *column\_name* IN (*SELECT STATEMENT*);

## Toán tử BETWEEN SQL

Toán tử BETweEN chọn các giá trị trong một phạm vi nhất định. Các giá trị có thể là số, văn bản hoặc ngày.

Toán tử BETweEN được bao gồm: bao gồm các giá trị bắt đầu và kết thúc.

SELECT column\_name(s)  
FROM table\_name  
WHERE column\_name BETWEEN value1 AND value2;

### **Thí dụ**

SELECT \* FROM Orders  
WHERE OrderDate BETWEEN #01/07/1996# AND #31/07/1996#;

HOẶC LÀ:

### **Thí dụ**

SELECT \* FROM Orders  
WHERE OrderDate BETWEEN '1996-07-01' AND '1996-07-31';

**Giá trị ngày tháng ta có thể để trong dấu ‘ ’ hoặc # #**

## JOIN SQL

Mệnh đề JOIN được sử dụng để kết hợp các hàng từ hai hoặc nhiều bảng, dựa trên một cột có liên quan giữa chúng.

### **Thí dụ**

SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate  
FROM Orders  
INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID;

* Với INNER JOIN  trong quá trình thực hiện tích hai bảng nó sẽ kiểm tra điều kiện ở ON (tức là sinhvien.LopID = lop.LopID), nếu đúng thì được chọn và sai thì bỏ qua
* INNER JOIN giống như việc thêm các giá trị của một bảng khác vào một bảng khác.
* **(INNER)** JOIN: Trả về các bản ghi có giá trị khớp trong cả hai bảng
* **LEFT (OUTER)** JOIN: Trả lại tất cả các bản ghi từ bảng bên trái và các bản ghi khớp từ bảng bên phải
* **RIGHT (OUTER)** JOIN: Trả lại tất cả các bản ghi từ bảng bên phải và các bản ghi khớp từ bảng bên trái
* **FULL (OUTER)** JOIN: Trả lại tất cả các bản ghi khi có một trận đấu trong bảng bên trái hoặc bên phải

      

Lệnh INNER JOIN tên cột ON Điều kiện join 2 bảng lại với nhau với điều kiện sau phần ON. Lệnh INNER JOIN sẽ nhanh hơn nhiều so với cách thông thường

**Sử dụng phép tích đề cát**

|  |  |
| --- | --- |
| 1  2  3 | SELECT \*  FROM sinhvien, lop  WHERE sinhvien.LopID = lop.LopID |

**Sử dụng INNER JOIN**

|  |  |
| --- | --- |
| 1  2 | SELECT \*  FROM sinhvien INNER JOIN lop ON sinhvien.LopID = lop.LopID |

## SQL RIGHT THAM GIA Từ khóa

Từ khóa RIGHT THAM GIA trả về tất cả các bản ghi từ bảng bên phải (bảng2) và các bản ghi khớp từ bảng bên trái (bảng1). Kết quả là NULL từ phía bên trái, khi không có kết quả khớp.

### **Cú pháp tham gia**

SELECT column\_name(s)  
FROM table1  
RIGHT JOIN table2ON table1.column\_name = table2.column\_name;

**Lưu ý:** Trong một số cơ sở dữ liệu, RIGHT THAM GIA được gọi là RIGHT OUTER THAM GIA.

## SQL TRÁI tham gia từ khóa

Từ khóa LEFT THAM GIA trả về tất cả các bản ghi từ bảng bên trái (bảng1) và các bản ghi khớp từ bảng bên phải (bảng2). Kết quả là NULL từ phía bên phải, nếu không có kết quả khớp.

### **Cú pháp tham gia**

SELECT column\_name(s)  
FROM table1  
LEFT JOIN table2ON table1.column\_name = table2.column\_name;

**Lưu ý:** Trong một số cơ sở dữ liệu, LEFT THAM GIA được gọi là LEFT OUTER THAM GIA.

**Khi ta sử dụng lệnh INNER JOIN csdl sẽ gộp 2 bảng vào thành 1, trong đó dữ liệu sẽ được sắp xếp theo bảng thứ 2. Và khi đó ta chỉ gọi các các đối tượng trong bảng 1 ra là sẽ có cả dữ liệu của bảng 2**

## SQL tự THAM GIA

Tự THAM GIA là một tham gia thường xuyên, nhưng bảng được tham gia với chính nó.

### **Cú pháp tự THAM GIA**

SELECT column\_name(s)  
FROM table1 T1, table1 T2  
WHERE condition;

## Toán tử SQL UNION

Toán tử UNION được sử dụng để kết hợp tập kết quả của hai hoặc nhiều câu lệnh CHỌN.

* Mỗi câu lệnh CHỌN trong UNION phải có cùng số cột
* Các cột cũng phải có kiểu dữ liệu tương tự
* Các cột trong mỗi câu lệnh CHỌN cũng phải theo cùng một thứ tự

### **Cú pháp của UNION**

SELECT column\_name(s) FROM table1  
UNION  
SELECT column\_name(s) FROM table2;

Toán tử UNION chỉ chọn các giá trị riêng biệt theo mặc định. Để cho phép các giá trị trùng lặp, sử dụng UNION ALL:

SELECT *column\_name(s)* FROM *table1*  
UNION ALL  
SELECT *column\_name(s)* FROM *table2*;

**Lưu ý:** Tên cột trong tập kết quả thường bằng với tên cột trong câu lệnh CHỌN đầu tiên trong UNION.