BÀI THỰC HÀNH SỐ 2:
Thao tác dữ liệu với T-SQL cơ bản

Mục lục

1.Structured Query Language (SQL)

1.1 SQL là ngôn ngữ của cơ sở dữ liệu quan hệ

SQL, viết tắt của Structured Query Language (ngôn ngữ hỏi có cấu trúc), là công cụ sử dụng để tổ chức, quản lý và truy xuất dữ liệu được lưu trữ trong các cơ sở dữ liệu. SQL là một hệ thống ngôn ngữ bao gồm tập các câu lệnh sử dụng để tương tác với cơ sở dữ liệu quan hệ.

Khả năng của SQL vượt xa so với một công cụ truy xuất dữ liệu, mặc dù đây là mục đích ban đầu khi SQL được xây dựng nên và truy xuất dữ liệu vẫn còn là một trong những chức năng quan trọng của nó. SQL được sử dụng để điều khiển tất cả các chức năng mà một hệ quản trị cơ sở dữ liệu cung cấp cho người dùng bao gồm:

Định nghĩa dữ liệu: SQL cung cấp khả năng định nghĩa các cơ sở dữ liệu, các cấu trúc lưu trữ và tổ chức dữ liệu cũng như mối quan hệ giữa các thành phần dữ liệu.

Truy xuất và thao tác dữ liệu: Với SQL, người dùng có thể dễ dàng thực hiện các thao tác truy xuất, bổ sung, cập nhật và loại bỏ dữ liệu trong các cơ sở dữ liệu.

Điều khiển truy cập: SQL có thể được sử dụng để cấp phát và kiểm soát các thao tác của người sử dụng trên dữ liệu, đảm bảo sự an toàn cho cơ sở dữ liệu

Đảm bảo toàn vẹn dữ liệu: SQL định nghĩa các ràng buộc toàn vẹn trong cơ sở dữ liệu nhờ đó đảm bảo tính hợp lệ và chính xác của dữ liệu trước các thao tác cập nhật cũng như các lỗi của hệ thống.

Như vậy, có thể nói rằng SQL là một ngôn ngữ hoàn thiện được sử dụng trong các hệ thống cơ sở dữ liệu và là một thành phần không thể thiếu trong các hệ quản trị cơ sở dữ liệu. Mặc dù SQL không phải là một ngôn ngữ lập trình như C, C++, Java,... song các câu lệnh mà SQL cung cấp có thể được nhúng vào trong các ngôn ngữ lập trình nhằm xây dựng các ứng dụng tương tác với cơ sở dữ liệu.

Khác với các ngôn ngữ lập trình quen thuộc như C, C++, Java,... SQL là ngôn ngữ có tính khai báo. Với SQL, người dùng chỉ cần mô tả các yêu cầu cần phải thực hiện trên cơ sở dữ liệu mà không cần phải chỉ ra cách thức thực hiện các yêu cầu như thế nào. Chính vì vậy, SQL là ngôn ngữ dễ tiếp cận và dễ sử dụng.

1.2 Vai trò của SQL

Bản thân SQL không phải là một hệ quản trị cơ sở dữ liệu, nó không thể tồn tại độc lập. SQL thực sự là một phần của hệ quản trị cơ sở dữ liệu, nó xuất hiện trong các hệ quản trị cơ sở dữ liệu với vai trò ngôn ngữ và là công cụ giao tiếp giữa người sử dụng và hệ quản trị cơ sở dữ liệu.

Trong hầu hết các hệ quản trị cơ sở dữ liệu quan hệ, SQL có những vai trò như sau:

SQL là ngôn ngữ hỏi có tính tương tác: Người sử dụng có thể dễ dàng thông qua các trình tiện ích để gởi các yêu cầu dưới dạng các câu lệnh SQL đến cơ sở dữ liệu và nhận kết quả

trả về từ cơ sở dữ liệu

SQL là ngôn ngữ lập trình cơ sở dữ liệu: Các lập trình viên có thể nhúng các câu lệnh SQL vào trong các ngôn ngữ lập trình để xây dựng nên các chương trình ứng dụng giao tiếp với cơ sở dữ liệu

SQL là ngôn ngữ quản trị cơ sở dữ liệu: Thông qua SQL, người quản trị cơ sở dữ liệu có thể quản lý được cơ sở dữ liệu, định nghĩa các cấu trúc lưu trữ dữ liệu, điều khiển truy cập cơ sở dữ liệu,...

SQL là ngôn ngữ cho các hệ thống khách/chủ (client/server): Trong các hệ thống cơ sở dữ liệu khách/chủ, SQL được sử dụng như là công cụ để giao tiếp giữa các trình ứng dụng phía máy khách với máy chủ cơ sở dữ liệu.

SQL là ngôn ngữ truy cập dữ liệu trên Internet: Cho đến nay, hầu hết các máy chủ Web cũng như các máy chủ trên Internet sử dụng SQL với vai trò là ngôn ngữ để tương tác với dữ liệu trong các cơ sở dữ liệu.

SQL là ngôn ngữ cơ sở dữ liệu phân tán: Đối với các hệ quản trị cơ sở dữ liệu phân tán, mỗi một hệ thống sử dụng SQL để giao tiếp với các hệ thống khác trên mạng, gởi và nhận các yêu cầu truy xuất dữ liệu với nhau.

SQL là ngôn ngữ sử dụng cho các cổng giao tiếp cơ sở dữ liệu: Trong một hệ thống mạng máy tính với nhiều hệ quản trị cơ sở dữ liệu khác nhau, SQL thường được sử dụng như là một chuẩn ngôn ngữ để giao tiếp giữa các hệ quản trị cơ sở dữ liệu.

1.3 Giới thiệu sơ lược về Transact SQL (T-SQL)

Transact-SQL là ngôn ngữ SQL mở rộng dựa trên SQL chuẩn của ISO (International Organization for Standardization) và ANSI (American National Standards Institute) được sử dụng trong SQL Server khác với P-SQL (Procedural-SQL) dùng trong Oracle.

SQL chuẩn bao gồm khoảng 40 câu lệnh. Trong các hệ quản trị cơ sở dữ liệu khác nhau, mặc dù các câu lệnh đều có cùng dạng và cùng mục đích sử dụng song mỗi một hệ quản trị cơ sở dữ liệu có thể có một số thay đổi nào đó. Điều này đôi khi dẫn đến cú pháp chi tiết của các câu lệnh có thể sẽ khác nhau trong các hệ quản trị cơ cơ sở dữ liệu khác nhau.

T-SQL được chia làm 3 nhóm:

1.3.1 Ngôn ngữ định nghĩa dữ liệu (Data Definition Language – DDL)

Đây là những lệnh dùng để tạo (create), thay đổi (alter) hay xóa (drop) các đối tượng trong CSDL. Các câu lệnh DDL thường có dạng:

Create object

Alter object

Drop object

Trong đó object có thể là: table, view, storedprocedure, function, trigger…

Ví dụ: Câu lệnh **Create** sau sẽ tạo một bảng mới có tên là CAULACBO trong CSDL QLBongda. Bảng CAULACBO này gồm có ba cột: MACLB, TENCLB, MASAN, MATINH

Lưu ý: Nếu trong SQL Server 2005 chưa có CSDL QLBongda, hãy tạo một CSDL có tên QLBongda theo hướng dẫn trong Chương 1.

```
create table CAULACBO
(
    MACLB varchar(5) primary key,
    TENCLB  nvarchar(100) not null,
    MASAN varchar(5) not null
)
```

Để chạy câu lệnh SQL trên, mở một ***Query Editor***, copy câu lệnh vào ***Query Editor***, bôi đen toàn bộ câu lệnh và bấm **F5**.

Tiếp theo, dùng lệnh **alter** để thay đổi cấu trúc bảng CAULACBO. Cụ thể là một thêm một cột mới có tên MATINH vào bảng CAULACBO.

```
alter table CAULACBO1
add MATINH  varchar(5) not null
```

Cuối cùng, dùng lệnh **drop** để xóa hoàn toàn bảng CAULACBO ra khỏi CSDL, nghĩa là toàn bộ định nghĩa bảng và các dữ liệu bên trong đều bị xóa.

```
drop table CAULACBO1
```

Lưu ý: Lệnh drop khác với lệnh delete. Lệnh delete chỉ xóa các dòng dữ liệu có trong bảng

1.3.2 Ngôn ngữ điều khiển dữ liệu (Data control language – DCL)

Đây là các lệnh quản lý quyền truy cập lên các object (table, view, storedprocedure…). Bao gồm:

Grant
Deny
Revoke

Ví dụ: Lệnh **grant** sẽ cấp quyền **Select** trên bảng CAULACBO trong CSDL QLBongda cho các **Users** thuộc Role public

```
grant select on CAULACBO
to public
```

Sau khi thực hiên lệnh này, có Users trong Role public có thể thực hiện câu lệnh Select trên bảng CAULACBO trong CSDL QLBongda.

Dùng lệnh **deny** để từ chối quyền select trên bảng CAULACBO trong CSDL QLBongda của

các Users thuộc Role public

```
deny select on caulacbo to public
```

Sau khi thực hiện lệnh này, có Users trong Role public sẽ không thể thực hiện câu lệnh Select trên bảng CAULACBO trong CSDL QLBongDa.

Dùng lệnh **revoke** để xóa bỏ các quyền được cấp hay từ chối trước đó.

```
revoke select on caulacbo to public
```

Sau khi thực hiện lệnh này, các quyền được gán hay từ chối của Users trong Role public trên bảng CAULACBO trong CSDL QLBongDa sẽ được "xóa" hoàn toàn.

1.3.3 Ngôn ngữ thao tác dữ liệu (Data manipulation language – DML

Đây là các lệnh phổ biến dùng để xử lý dữ liệu. Bao gồm:

Select

Insert

Update

Delete

Ví dụ: a) Câu lệnh sau sẽ lọc ra các câu lạc bộ có tên bắt đầu bằng chữ A trong bảng CAULACBO

```
select *
from CAULACBO as CLB
where CLB.TENCLB like 'T%'
```

Kết quả hiển thị

1	TPY	Thép Phú Yên	TH	PY

Dấu * hàm ý là lựa chọn tất cả các cột của bảng Nhanvien. Toán tử like và ký tự đại diện sẽ được nói trong phần sau.

b) Thực hiện thêm dữ liệu về một câu lạc bộ mới vào trong bảng CAULACBO

```
insert into CAULACBO
values(' SLNA', N'Sông Lam Nghệ An','TH','PY')
```

```
Kết quả:
 (1 row(s) affected)
```

c) Thực hiện cập nhật lai mã tinh của câu lạc bộ có maclb là SLNA

```
update CAULACBO
set MATINH = 'GL'

where maclb like 'SLNA'
```

Kết quả:

```
(1 row(s) affected)
```

d) Thực hiện sẽ xóa thông tin của câu lạc bộ có maclb là SLNA trong bảng CAULACBO

```
delete CAULACBO where maclb = 'SLNA'
```

```
Kết quả:
(1 row(s) affected)
```

1.3.4 Cú pháp của T-SQL

Các đối tượng trong cơ sở dữ liệu dựa trên SQL (table, view, index, storedprocedure…) được xác định thông qua tên của đối tượng (hay còn gọi là identifier). Tên của các đối tượng là duy nhất trong mỗi cơ sở dữ liệu. Tên được sử dụng nhiều nhất trong các truy vấn SQL và được xem là nền tảng trong cơ sở dữ liệu quan hệ là tên bảng và tên cột.

Có hai loại Identifiers một loại thông thường (Regular Identifier) và một loại gọi là Delimited Identifier, loại này cần có dấu "" hay dấu [] để ngăn cách. Loại Delimited được dùng đối với các chữ trùng với từ khóa của SQL Server (reserved keyword) hay các chữ có khoảng trống.

Ví dụ:

*Select **

From "My table"

Where [sum] = 10

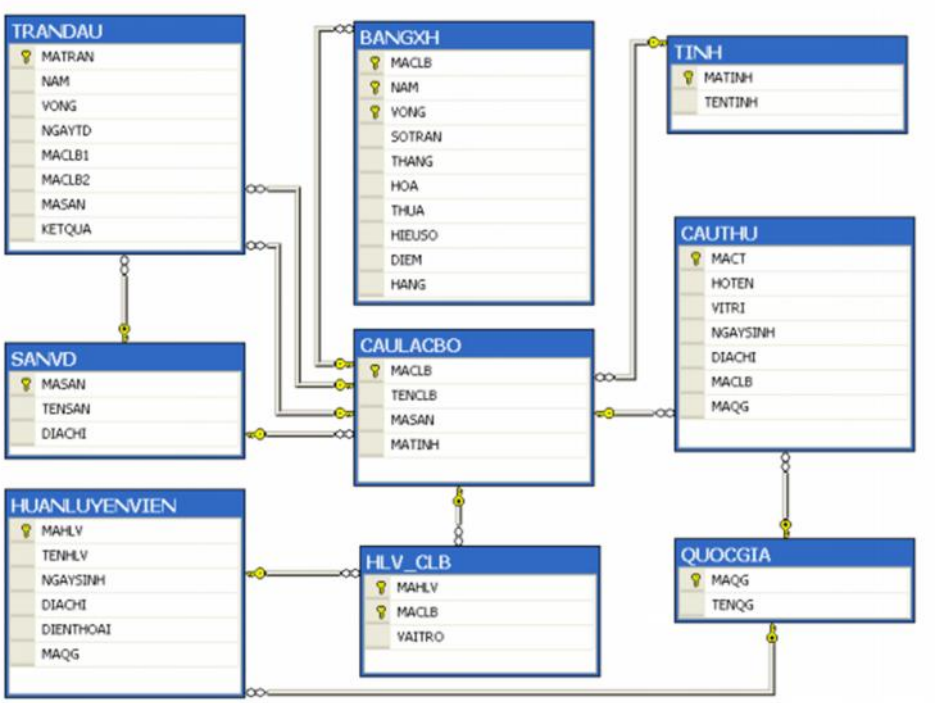
Trong các cơ sở dữ liệu lớn với nhiều người sử dụng, khi ta chỉ định tên của một bảng nào đó trong câu lệnh SQL, hệ quản trị cơ sở dữ liệu hiểu đó là tên của bảng do ta sở hữu (tức là bảng do ta tạo ra). Thông thường, trong các hệ quản trị cơ sở dữ liệu này cho phép những người dùng khác nhau tạo ra những bảng trùng tên với nhau mà không gây ra xung đột về tên. Nếu trong một câu lệnh SQL ta cần chỉ đến một bảng do một người dùng khác sở hữu (hiển nhiên là phải được phép) thì tên của bảng phải được viết sau tên của người sở hữu và phân cách với tên người sở hữu bởi dấu chấm:

tên_người_sở_hữu.tên_bảng

Một số đối tượng cơ sở dữ liệu khác (như khung nhìn, thủ tục, hàm), việc sử dụng tên cũng tương tự như đối với bảng.

Ta có thể sử dụng tên cột một cách bình thường trong các câu lệnh SQL bằng cách chỉ cần chỉ định tên của cột trong bảng. Tuy nhiên, nếu trong câu lệnh có liên quan đến hai cột trở lên có cùng tên trong các bảng khác nhau thì bắt buộc phải chỉ định thêm tên bảng trước tên cột; tên bảng và tên cột được phân cách nhau bởi dấu chấm

Ví dụ: Giả sử chúng ta có CSDL như sau:



Để tìm ra thông tin cầu thủ ‘’ đá cho câu lạc bộ nào, ở vị trí nào, câu truy vấn như sau:

```
select HOTEN,VITRI,TENCLB
from CAUTHU,CAULACBO
where (CAUTHU.MACLB=CAULACBO.MACLB)and (HOTEN = N'Nguyễn Công Vinh')
```

1	Nguyễn Công Vinh	Tiền Đạo	Hoàng Anh Gia Lai

1.3.5 Các kiểu dữ liệu

Bảng dưới đây liệt kê một số kiểu dữ liệu thông dụng được sử dụng trong SQL.

Char(n)	Kiểu chuỗi với độ dài cố định
Nchar(n)	Kiểu chuỗi với độ dài cố định hỗ trợ UNICODE
Varchar(n)	Kiểu chuỗi với độ dài chính xác
Nvarchar(n)	Kiểu chuỗi với độ dài chính xác hỗ trợ UNICODE
Int	Số nguyên có giá trị từ -2^{31} đến $2^{31} - 1$
Tinyint	Số nguyên có giá trị từ 0 đến 255.
Smallint	Số nguyên có giá trị từ -2^{15} đến $2^{15} - 1$
Bigint	Số nguyên có giá trị từ -263 đến 263-1
Numeric	Kiểu số với độ chính xác cố định.
Decimal	Tương tự kiểu Numeric
Float	Số thực có giá trị từ -1.79E+308 đến 1.79E+308
Real	Số thực có giá trị từ -3.40E + 38 đến 3.40E + 38
Money	Kiểu tiền tệ
Bit	Kiểu bit (có giá trị 0 hoặc 1)
Datetime	Kiểu ngày giờ (chính xác đến phần trăm của giây)
Smalldatetime	Kiểu ngày giờ (chính xác đến phút)
Binary	Dữ liệu nhị phân với độ dài cố định (tối đa 8000 bytes)
Varbinary	Dữ liệu nhị phân với độ dài chính xác (tối đa 8000 bytes)
Image	Dữ liệu nhị phân với độ dài chính xác (tối đa 2,147,483,647 bytes)
Text	Dữ liệu kiểu chuỗi với độ dài lớn (tối đa 2,147,483,647 ký tự)
Ntext	Dữ liệu kiểu chuỗi với độ dài lớn và hỗ trợ UNICODE (tối đa 1,073,741,823 ký tự)

Ví dụ: Mỗi cột trong bảng sẽ chứa những dữ liệu thuộc về duy nhất một kiểu dữ liệu trong SQL Server. Cột nào chứa những dữ liệu thuộc kiểu nào sẽ được quy định lúc định nghĩa bảng.

```
CREATE TABLE [dbo].[CAUTHU](
	[MACT] [numeric](18, 0) IDENTITY(1,1) PRIMARY KEY NOT NULL,
	[HOTEN] [nvarchar](100) NOT NULL,
	[VITRI] [nvarchar](50) NOT NULL,
	[NGAYSINH] [datetime] NULL,
	[DIACHI] [nvarchar](200) NULL,
	[MACLB] [varchar](5) NOT NULL,
	[MAQG] [varchar](5) NOT NULL,
	[SO] [int] NOT NULL,

)
```

1.3.6 Biến (Variables)

Biến trong T-SQL cũng có chức năng tương tự như trong các ngôn ngữ lập trình khác nghĩa là cần khai báo trước loại dữ liệu trước khi sử dụng. Biến được bắt đầu bằng dấu @ (Đối với các biến toàn cục - global variable - thì có hai dấu @@)

Ví dụ: Ví dụ dưới đây khai báo một biến có tên @SLCauLacBo thông qua từ khóa **declare**. Biến này lưu số câu lạc bộ đếm được thông qua hàm count. Sau đó in ra giá trị của biến.

```
declare @SLCauLacBo int
select @SLCauLacBo= count(*)
from CAULACBO

print @SLCauLacBo
```

1.3.7 Hàm (Function)

Có 2 loại hàm: một loại là được xây dựng sẵn trong SQL Server 2005 Edition (built-in) và một loại do người dùng tự định nghĩa (user-defined) Các hàm Built-In được chia làm 3 nhóm:

Rowset Functions: Loại này thường trả về một *object* và được đối xử như một *table*. Ví dụ như hàm **OPENQUERY** sẽ trả về một *recordset* và có thể đứng vị trí của một *table* trong câu lệnh **Select**.

Aggregate Functions: Loại này làm việc trên một số giá trị và trả về một giá trị đơn hay là các giá trị tổng. Ví dụ như hàm ***AVG*** sẽ trả về giá trị trung bình của một cột.

Scalar Functions: Loại này làm việc trên một giá trị đơn và trả về một giá trị đơn. Trong loại này lại chia làm nhiều loại nhỏ như các hàm về toán học, về thời gian, xử lý kiểu dữ liệu String....Ví dụ như hàm *MONTH('2002-09-30')* sẽ trả về tháng 9.

Các hàm User-Defined (được tạo ra bởi câu lệnh **CREATE FUNCTION** và phần body thường được gói trong cặp lệnh **BEGIN...END**) cũng được chia làm các nhóm như sau:

Scalar Functions: Loại này cũng trả về một giá trị đơn bằng câu lệnh RETURNS.

Table Functions : Loại này trả về một table

1.3.8 Các toán tử (Operators)

Trong SQL Server các biểu diễn (expression) có thể xuất hiện nhiều toán tử. Độ ưu tiên của toán tử sẽ quyết định thứ tự thực hiện của các phép tính. Thứ tự thực hiện ảnh hưởng rất lớn đến kết quả.

Bảng dưới đây mô tả các toán tử trong SQL Server 2005 Standard Edititon và mức độ ưu tiên của các toán tử đó.

Level	Operators
1	* (Multiply), / (Division), % (Modulo)
2	+ (Positive), - (Negative), + (Add), (+ Concatenate), - (Subtract),
3	=, >, <, >=, <=, <>, !=, !>, !< (Comparison operators)
4	NOT
5	AND

6	ALL, ANY, BETWEEN, IN, LIKE, OR, SOME
7	= (Assignment)

1.3.9 Các thành phần điều khiển (Control of flow)

Như BEGIN...END, BREAK, CONTINUE, GOTO, IF...ELSE, RETURN, WHILE…

1.3.10 Chú thích (Comment)

T-SQL dùng kí hiệu -- để chú thích cho một dòng đơn và kí hiệu /*…*/ để chú thích cho một nhóm dòng

Ví dụ:

/* Minh họa chú thích

Chú thích cho một dòng đơn và một nhóm các dòng*/

```
DECLARE @MyNumber int   -- khai báo biến
SET @MyNumber = 4 - 2 + 27
-- kết quả là 29

SELECT @MyNumber
```

1.3.11 Giá trị NULL

Một cơ sở dữ liệu là sự phản ánh của một hệ thống trong thế giới thực, do đó các giá trị dữ liệu tồn tại trong cơ sở dữ liệu có thể không xác định được. Một giá trị không xác định được xuất hiện trong cơ sở dữ liệu có thể do một số nguyên nhân sau:

- Giá trị đó có tồn tại nhưng không biết.
- Không xác định được giá trị đó có tồn tại hay không.
- Tại một thời điểm nào đó giá trị chưa có nhưng rồi có thể sẽ có.
- Giá trị bị lỗi do tính toán (tràn số, chia cho không,...)

Những giá trị không xác định được biểu diễn trong cơ sở dữ liệu quan hệ bởi các giá trị NULL. Đây là giá trị đặc biệt và không nên nhầm lẫn với chuỗi rỗng (đối với dữ liệu kiểu chuỗi) hay giá trị không (đối với giá trị kiểu số). Giá trị NULL đóng một vai trò quan trọng trong các cơ sở dữ liệu và hầu hết các hệ quản trị cơ sở dữ liệu quan hệ hiện nay đều hỗ trợ việc sử dụng giá trị này.

2.Ngôn ngữ thao tác dữ liệu – DML

SQL được xem như là công cụ hữu hiệu để thực hiện các yêu cầu truy vấn và thao tác trên dữ liệu. Trong chương này, ta sẽ bàn luận đến nhóm các câu lệnh trong SQL được sử dụng cho mục đích này. Nhóm các câu lệnh này được gọi chung là ngôn ngữ thao tác dữ liệu (DML: Data Manipulation Language) bao gồm các câu lệnh sau:
SELECT: Sử dụng để truy xuất dữ liệu từ môt hoặc nhiều bảng.

INSERT: Thêm dữ liệu.

UPDATE: Cập nhật dữ liệu

DELETE: Xoá dữ liệu
Trong số các câu lệnh này, có thể nói **SELECT** là câu lệnh tương đối phức tạp và được sử dụng nhiều trong cơ sở dữ liệu. Với câu lệnh này, ta không chỉ thực hiện các yêu cầu truy xuất dữ liệu đơn thuần mà còn có thể thực hiện được các yêu cầu thống kê dữ liệu phức tạp. Cũng chính vì vậy, phần đầu của chương này sẽ tập trung tương đối nhiều đến câu lệnh **SELECT**. Các câu lệnh **INSERT, UPDATE và DELETE** được bàn luận đến ở cuối chương

2.1 Câu lệnh SELECT

Câu lệnh SELECT được sử dụng để truy xuất dữ liệu từ các dòng và các cột của một hay nhiều bảng, khung nhìn. Câu lệnh này có thể dùng để thực hiện phép chọn (tức là truy xuất một tập con các dòng trong một hay nhiều bảng), phép chiếu (tức là truy xuất một tập con các cột trong một hay nhiều bảng) và phép nối (tức là liên kết các dòng trong hai hay nhiều bảng để truy xuất dữ liệu). Ngoài ra, câu lệnh này còn cung cấp khả năng thực hiện các thao tác truy vấn và thống kê dữ liệu phức tạp khác.
Cú pháp chung của câu lệnh SELECT có dạng:

SELECT [ALL | DISTINCT][TOP n] danh_sách_chọn [INTO tên_bảng_mới]
FROM danh_sách_bảng/khung_nhìn

[WHERE điều_kiện] [GROUP BY danh_sách_cột] [HAVING điều_kiện]
[ORDER BY cột_sắp_xếp]

[COMPUTE danh_sách_hàm_gộp [BY danh_sách_cột]]

Điều cần lưu ý đầu tiên đối với câu lệnh này là các thành phần trong câu lệnh SELECT nếu được sử dụng phải tuân theo đúng thứ tự như trong cú pháp. Nếu không, câu lệnh sẽ được xem là không hợp lệ.

Câu lệnh SELECT được sử dụng để tác động lên các bảng dữ liệu và kết quả của câu lệnh cũng được hiển thị dưới dạng bảng, tức là một tập hợp các dòng và các cột (ngoại trừ trường hợp sử dụng câu lệnh SELECT với mệnh đề COMPUTE).

Ví dụ dưới đây hiển thị tên cầu thủ và vị trí cầu thủ đó trong bảng

```
select hoten, vitri
from cauthu
```

	hoten	vitri
1	Nguyễn Vũ Phong	Tiền Đạo
2	Nguyễn Công Vinh	Tiền Đạo
3	Trần Tấn Tài	Tiền vệ
4	Phan Hồng Sơn	Thủ môn
5	Ronaldo	Tiền vệ
6	Robinho	Tiền vệ
7	Vidic	Hậu vệ
8	Trần Văn Santos	Thủ môn

2.1.1 Danh sách chọn trong câu lệnh SELECT

Danh sách chọn trong câu lệnh SELECT được sử dụng để chỉ định các trường, các biểu thức cần hiển thị trong các cột của kết quả truy vấn. Các trường, các biểu thức được chỉ định ngay sau từ khoá SELECT và phân cách nhau bởi dấu phẩy. Sử dụng danh sách chọn trong câu lệnh SELECT bao gồm các trường hợp sau:

Chọn tất cả các cột: Như đã nói trong chương 1, chúng ta dùng dấu * trong câu lệnh Select để hàm ý chọn hết tất cả các cột. Trong trường hợp này, các cột được hiển thị trong kết quả truy vấn sẽ tuân theo thứ tự mà chúng đã được tạo ra khi bảng được định nghĩa.

Ví dụ:

```
select *
from cauthu
```

	MACT	HOTEN	VITRI	NGAYSINH	DIACHI	MACLB	MA...	SO
1	1	Nguyễn Vũ Phong	Tiền Đạo	1990-02-20 00:00:00.000	NULL	BBD	VN	10
2	2	Nguyễn Công Vinh	Tiền Đạo	1992-03-10 00:00:00.000	NULL	HAGL	VN	9
3	4	Trần Tấn Tài	Tiền vệ	1989-11-12 00:00:00.000	NULL	BBD	VN	8
4	5	Phan Hồng Sơn	Thủ môn	1991-06-10 00:00:00.000	NULL	HAGL	VN	1
5	6	Ronaldo	Tiền vệ	1989-12-12 00:00:00.000	NULL	SDN	BRA	7
6	7	Robinho	Tiền vệ	1989-10-12 00:00:00.000	NULL	SDN	BRA	8
7	8	Vidic	Hậu vệ	1987-10-15 00:00:00.000	NULL	HAGL	ANH	3
8	9	Trần Văn Santos	Thủ môn	1990-10-21 00:00:00.000	NULL	BBD	TBN	1
9	10	Nguyễn Trường	Hậu vệ	1993-08-26 00:00:00.000	NULL	BBD	VN	4

Chọn một số cột cụ thể: Trong trường hợp cần chỉ định cụ thể các cột cần hiển thị trong kết quả truy vấn, ta chỉ định danh sách các tên cột trong danh sách chọn. Thứ tự của các cột trong kết quả truy vấn tuân theo thứ tự của các trường trong danh sách chọn.

Ví dụ:

```
select HOTEN, VITRI
from cauthu
```

Lưu ý: Nếu truy vấn được thực hiện trên nhiều bảng/khung nhìn và trong các bảng/khung nhìn có các trường trùng tên thì tên của những trường này nếu xuất hiện trong danh sách chọn phải được viết dưới dạng: tên_bảng.tên_trường

Thay đổi tiêu đề các cột:

Trong kết quả truy vấn, tiêu đề của các cột mặc định sẽ là tên của các trường tương ứng trong bảng. Tuy nhiên, để các tiêu đề trở nên thân thiện hơn, ta có thể đổi tên các tiêu đề của các cột. Để đặt tiêu đề cho một cột nào đó, ta sử dụng cách viết:

tiêu_đề_cột = tên_trường hoặc *tên_trường AS tiêu_đề_cột* hoặc *tên_trường tiêu_đề_cột*

Ví dụ:

```
select [Mã cầu thủ]=MACT, HOTEN AS [Tên cầu thủ], VITRI [vị trí]
from CAUTHU
```

	Mã cầu t...	Tên cầu thủ	vị trí
1	1	Nguyễn Vũ Phong	Tiền Đạo
2	2	Nguyễn Công Vinh	Tiền Đạo
3	4	Trần Tấn Tài	Tiền vệ
4	5	Phan Hồng Sơn	Thủ môn
5	6	Ronaldo	Tiền vệ
6	7	Robinho	Tiền vệ
7	8	Vidic	Hậu vệ
8	9	Trần Văn Santos	Thủ môn
9	10	Nguyễn Trường	Hậu vệ

Sử dụng cấu trúc CASE…WHEN:

Cấu trúc CASE được sử dụng trong danh sách chọn nhằm thay đổi kết quả của truy vấn tuỳ thuộc vào các trường hợp khác nhau. Cấu trúc này có cú pháp như sau:

CASE biểu_thức

WHEN biểu_thức_kiểm_tra THEN kết_quả

[...]

[ELSE kết_quả_của_else]

END hoặc: CASE

WHEN điều_kiện THEN kết_quả

[...]

[ELSE kết_quả_của_else]

END

Ví dụ: Câu lệnh SQL dưới đây sẽ hiện thị giới tính của khách hàng tùy theo giá trị thực được lưu trong CSDL. Nếu giá trị trong CSDL là FALSE-> hiện thị giới tính NỮ, nếu giá trị là TRUE-> hiện thị giới tính NAM.

```
select HOTEN, DIACHI, case GENDER
when 1 then 'NAM' else N'NỮ'
end as [GIỚI TÍNH]

from CAUTHU
```

Câu lệnh trên cũng có thể viết như sau:

Loại bỏ các dòng dữ liệu trùng nhau:

Từ khóa DISTINCT sẽ loại bỏ các dòng dữ liệu giống nhau.

Ví dụ: Hiển thị các Mã câu lạc bộ tham gia thi đấu ở sân nhà trong bảng TRANDAU

```
select distinct MACLB1
from TRANDAU
```

	MACLB1
1	BBD
2	KKH
3	SDN
4	TPY

Lựa chọn một số lượng giới hạn các dòng: Từ khóa **TOP n** sẽ trả về chỉ n dòng dữ liệu

Ví dụ: Hiển thị 7 cầu thủ từ bảng CAUTHU

```
select top 7 HOTEN AS [Tên cầu thủ]
from CAUTHU
```

	Tên cầu thủ
1	Nguyễn Vũ Phong
2	Nguyễn Công Vinh
3	Trần Tấn Tài
4	Phan Hồng Sơn
5	Ronaldo
6	Robinho
7	Vidic

Nếu sử dung TOP n PERCENT thì sẽ trả về n % số dòng dữ liệu hiện có trong CSDL.

2.1.2 Mệnh đề FROM

Mệnh đề FROM trong câu lệnh SELECT được sử dung nhằm chỉ định các bảng và khung nhìn cần truy xuất dữ liệu. Sau FROM là danh sách tên của các bảng và khung nhìn tham gia vào truy vấn, tên của các bảng và khung nhìn được phân cách nhau bởi dấu phẩy.

Ví dụ: Câu lệnh sau hiển thị thông tin cầu thủ

```
select *
from cauthu
```

	MACT	HOTEN	VITRI	NGAYSINH	DIACHI	MACLB	MA...	SO
1	1	Nguyễn Vũ Phong	Tiền Đạo	1990-02-20 00:00:00.000	NULL	BBD	VN	10
2	2	Nguyễn Công Vinh	Tiền Đạo	1992-03-10 00:00:00.000	NULL	HAGL	VN	9
3	4	Trần Tấn Tài	Tiền vệ	1989-11-12 00:00:00.000	NULL	BBD	VN	8
4	5	Phan Hồng Sơn	Thủ môn	1991-06-10 00:00:00.000	NULL	HAGL	VN	1
5	6	Ronaldo	Tiền vệ	1989-12-12 00:00:00.000	NULL	SDN	BRA	7
6	7	Robinho	Tiền vệ	1989-10-12 00:00:00.000	NULL	SDN	BRA	8
7	8	Vidic	Hậu vệ	1987-10-15 00:00:00.000	NULL	HAGL	ANH	3
8	9	Trần Văn Santos	Thủ môn	1990-10-21 00:00:00.000	NULL	BBD	TBN	1
9	10	Nguyễn Trường	Hậu vệ	1993-08-26 00:00:00.000	NULL	BBD	VN	4

Trong mệnh đề FROM có thể sử dụng bí danh (alias) nhằm làm cho câu truy vấn dễ nhìn
Ví dụ:

```
select HOTEN AS [Tên cầu thủ]
from CAUTHU ct

where ct.MACT = 4
```

2.1.3 Mệnh đề WHERE - điều kiện truy vấn dữ liệu

Mệnh đề WHERE trong câu lệnh SELECT được sử dụng nhằm xác định các điều kiện đối với việc truy xuất dữ liệu. Sau mệnh đề WHERE là một biểu thức logic và chỉ những dòng dữ liệu nào thoả mãn điều kiện được chỉ định mới được hiển thị trong kết quả truy vấn.

Ví dụ: Lọc ra thông tin các cầu thủ có mã cầu thủ >5

```
select HOTEN AS [Tên cầu thủ]
from CAUTHU ct

where ct.MACT > 2
```

	Tên cầu thủ
1	Trần Tấn Tài
2	Phan Hồng Sơn
3	Ronaldo
4	Robinho
5	Vidic
6	Trần Văn Santos
7	Nguyễn Trường Sơn

Trong mệnh đề WHERE thường sử dụng:

- Các toán tử kết hợp điều kiện (AND, OR)
- Các toán tử so sánh
- Kiểm tra giới hạn của dữ liệu (BETWEEN/ NOT BETWEEN)
- Tập hợp
- Kiểm tra khuôn dạng dữ liệu.
- Các giá trị NULL

Các toán tử so sánh

Toán tử	Ý nghĩa
=	Bằng
>	Lớn hơn
<	Nhỏ hơn
>=	Lớn hơn hoặc bằng
<=	Nhỏ hơn hoặc bằng
<>	Khác
!>	Không lớn hơn
!<	Không nhỏ hơn

Ví dụ: Hiển thị họ tên, ngày sinh theo định dạng dd/MM/yyyy và địa chỉ của các cầu thủ có tuổi lớn hơn 20

```
select      HOTEN,
convert (varchar, NGAYSINH, 103) as NGAYSINH, DIACHI
from CAUTHU

where year(getdate()) - year(NGAYSINH) > 20
```

	HOTEN	NGAYSINH	DIACHI
1	Nguyễn Vũ Phong	20/02/1990	NULL
2	Trần Tấn Tài	12/11/1989	NULL
3	Ronaldo	12/12/1989	NULL
4	Robinho	12/10/1989	NULL
5	Vidic	15/10/1987	NULL
6	Trần Văn Santos	21/10/1990	NULL

Kiểm tra giới hạn của dữ liệu

Để kiểm tra xem giá trị dữ liệu nằm trong (ngoài) một khoảng nào đó, ta sử dụng toán tử BETWEEN/ NOT BETWEEN như sau:

Mệnh đề	Ý nghĩa
variable BETWEEN a AND b	a <= variable <=b
variable NOT BETWEEN a AND b	variable <a hoặc variable > b

Ví dụ: Hiển thị họ tên, ngày sinh theo định dạng dd/MM/yyyy và địa chỉ của các cầu thủ có tuổi lớn hơn 20 và nhỏ hơn 30

```
select      HOTEN,
convert (varchar, NGAYSINH, 103) as NGAYSINH, DIACHI
from CAUTHU

where year(getdate()) - year(NGAYSINH) between 20 and 25
```

	HOTEN	NGAYSINH	DIACHI
1	Nguyễn Vũ Phong	20/02/1990	NULL
2	Trần Tấn Tài	12/11/1989	NULL
3	Phan Hồng Sơn	10/06/1991	NULL
4	Ronaldo	12/12/1989	NULL
5	Robinho	12/10/1989	NULL
6	Vidic	15/10/1987	NULL
7	Trần Văn Santos	21/10/1990	NULL

Toán tử làm việc trên tập hợp (IN/ NOT IN)

Từ khoá IN/ NOT IN được sử dụng khi ta cần chỉ định điều kiện tìm kiếm dữ liệu cho câu lệnh SELECT là một danh sách các giá trị. Sau IN/ NOT IN có thể là một danh sách các giá trị hoặc là một câu lệnh SELECT khác.

Ví dụ: Câu lệnh dưới đây lấy ra các thông tin của cầu thủ có mã là 5,6 hoặc 7

```
select      HOTEN,
convert (varchar, NGAYSINH, 103) as NGAYSINH, DIACHI
from CAUTHU

where MACT in (4,6,7)
```

	HOTEN	NGAYSINH	DIACHI
1	Trần Tấn Tài	12/11/1989	NULL
2	Ronaldo	12/12/1989	NULL
3	Robinho	12/10/1989	NULL

Ví dụ: Hiển thị các câu lạc bộ chưa thi đấu trận nào ở sân nhà. Sử dụng SELECT đ ứng sau mệnh đề IN/ NOT IN

```
select TENCLB
from CAULACBO
where MACLB not in

(Select MACLB1 from TRANDAU )
```

	TENCLB
1	Gạch Đồng Tâm Long An
2	Hoàng Anh Gia Lai

Toán tử LIKE/ NOT LIKE và ký tự đại diện (WildCard)

Từ khoá LIKE (NOT LIKE) sử dụng trong câu lệnh SELECT nhằm mô tả khuôn dạng của dữ liệu cần tìm kiếm. Chúng thường được kết hợp với các ký tự đại diện sau đây:

Ký tự đại diện	Ý nghĩa
%	Chuỗi ký tự bất kỳ gồm không hoặc nhiều ký tự
_	Một ký tự bất kì
[]	Một ký tự nằm trong giới hạn được chỉ định. Ví dụ:[a-f] hàm ý chỉ một trong các ký tự: a, b, c, d, e, f.
[^]	Một ký tự không nằm trong giới hạn được chỉ định. Ví dụ:[^a-f] hàm ý chỉ một ký tự khác tất cả các ký tự: a, b, c, d, e, f.

Ví dụ: Hiển thị các cầu thủ có tên bắt đầu bằng Nguyễn

```
select HOTEN
from CAUTHU

where HOTEN like N'Nguyễn %'
```

	HOTEN
1	Nguyễn Vũ Phong
2	Nguyễn Công Vinh
3	Nguyễn Trường Sơn

Giá trị NULL

Dữ liệu trong một cột cho phép NULL sẽ nhận giá trị NULL trong các trường hợp sau:

- Nếu không có dữ liệu được nhập cho cột và không có mặc định cho cột hay kiểu dữ liệu trên cột đó.
- Người sử dụng trực tiếp đưa giá trị NULL vào cho cột đó.
- Một cột có kiểu dữ liệu là kiểu số sẽ chứa giá trị NULL nếu giá trị được chỉ định gây tràn số.

Trong mệnh đề WHERE, để kiểm tra giá trị của một cột có giá trị NULL hay không, ta sử dụng cách viết:

```
WHERE tên_cột IS NULL
```

hoặc:

```
WHERE tên_cột IS NOT NULL
```

Ví dụ: Hiển thị các huấn luyện viên chưa nhập số điện thoại

```
select *
from HUANLUYENVIEN

where DIENTHOAI is NULL
```

	MAHLV	TENHLV	NGAYSINH	DIACHI	DIENTHOAI	MAQG
1	HL001	Vital	1955-10-15 00:00:00.000	NULL	NULL	BDN
2	HL004	Hoàng Anh Tuấn	1970-06-10 00:00:00.000	NULL	NULL	VN
3	HL006	Trần Văn Phúc	1965-03-02 00:00:00.000	NULL	NULL	VN

Tạo mới bảng dữ liệu từ câu lệnh SELECT

Câu lệnh SELECT ... INTO có tác dụng tạo một bảng mới có cấu trúc và dữ liệu được xác định từ kết quả của truy vấn. Bảng mới được tạo ra sẽ có số cột bằng số cột được chỉ định trong danh sách chọn và số dòng sẽ là số dòng kết quả của truy vấn

Ví dụ:

```
select      MACT,HOTEN,
convert (varchar, NGAYSINH, 103) as NGAYSINH, DIACHI into NEWTABLE

from CAUTHU
```

Lưu ý: Nếu trong danh sách chọn có các biểu thức thì những biểu thức này phải được đặt tiêu đề

Sắp xếp kết quả truy vấn

Mặc định, các dòng dữ liệu trong kết quả của câu truy vấn tuân theo thứ tự của chúng trong bảng dữ liệu hoặc được sắp xếp theo chỉ mục (nếu trên bảng có chỉ mục). Trong trường hợp muốn dữ liệu được sắp xếp theo chiều tăng hoặc giảm của giá trị của một hoặc nhiều trường, ta sử dụng thêm mệnh đề ORDER BY trong câu lệnh SELECT; Sau ORDER BY là danh sách các cột cần sắp xếp (tối đa là 16 cột). Dữ liệu được sắp xếp có thể theo chiều tăng (ASC) hoặc giảm (DESC), mặc định là sắp xếp theo chiều tăng. Nếu sau ORDER BY có nhiều cột thì việc sắp xếp dữ liệu sẽ được ưu tiên theo thứ tự từ trái qua phải.

Ví dụ: Sắp xếp thông tin cầu thủ theo thứ tự tuổi giảm dần

```
select HOTEN, year(getdate())- year(NGAYSINH) as AGE, VITRI
from CAUTHU
order by AGE DESC
```

	HOTEN	A...	VITRI
1	Vidic	24	Hậu vệ
2	Ronaldo	22	Tiền vệ
3	Robinho	22	Tiền vệ
4	Trần Tấn Tài	22	Tiền vệ
5	Nguyễn Vũ Phong	21	Tiền Đạo
6	Trần Văn Santos	21	Thủ môn
7	Phan Hồng Sơn	20	Thủ môn
8	Nguyễn Công Vinh	19	Tiền Đạo
9	Nguyễn Trường Sơn	18	Hậu vệ

Ta có thể chỉ định số thứ tự của cột cần được sắp xếp. Câu lệnh ở ví dụ trên có thể được viết lại như sau:

```
select HOTEN, year(getdate())- year(NGAYSINH) as AGE, VITRI
from CAUTHU
order by 2 DESC
```

2.1.4 Phép hợp (UNION)

Phép hợp được sử dụng trong trường hợp ta cần gộp kết quả của hai hay nhiều truy vấn thành một tập kết quả duy nhất. SQL cung cấp toán tử UNION để thực hiện phép hợp. Cú pháp như sau:

```
Câu_lệnh_1
UNION [ALL] Câu_lệnh_2
[UNION [ALL] Câu_lệnh_3]
...
[UNION [ALL] Câu_lệnh_n]
[ORDER BY        cột_sắp_xếp]
[COMPUTE         danh_sách_hàm_gộp [BY danh_sách_cột]]
```

Trong đó :

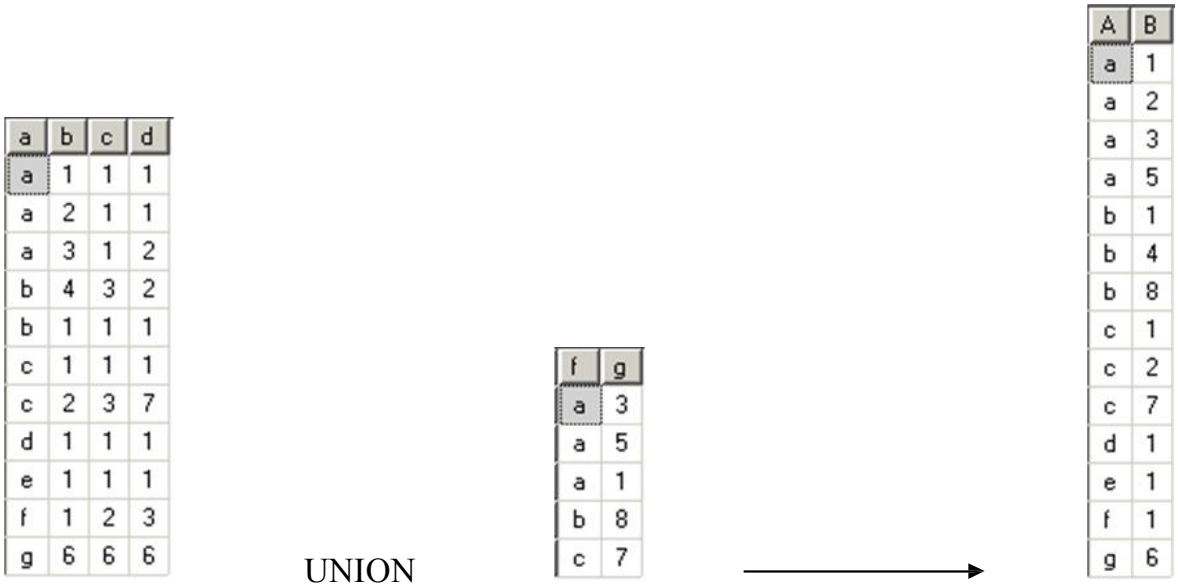
Câu_lệnh_1 có dạng

SELECT danh_sách_cột [INTO tên_bảng_mới]
[FROM danh_sách_bảng|khung_nhìn]
[WHERE điều_kiện] [GROUP BY danh_sách_cột] [HAVING điều_kiện]

và Câu_lệnh_i (i = 2,..,n) có dạng

SELECT danh_sách_cột
[FROM danh_sách_bảng|khung_nhìn]
[WHERE điều_kiện] [GROUP BY danh_sách_cột] [HAVING điều_kiện]

Ví dụ: Phép hợp giữa hai bảng dưới đây cho kết quả như sau:

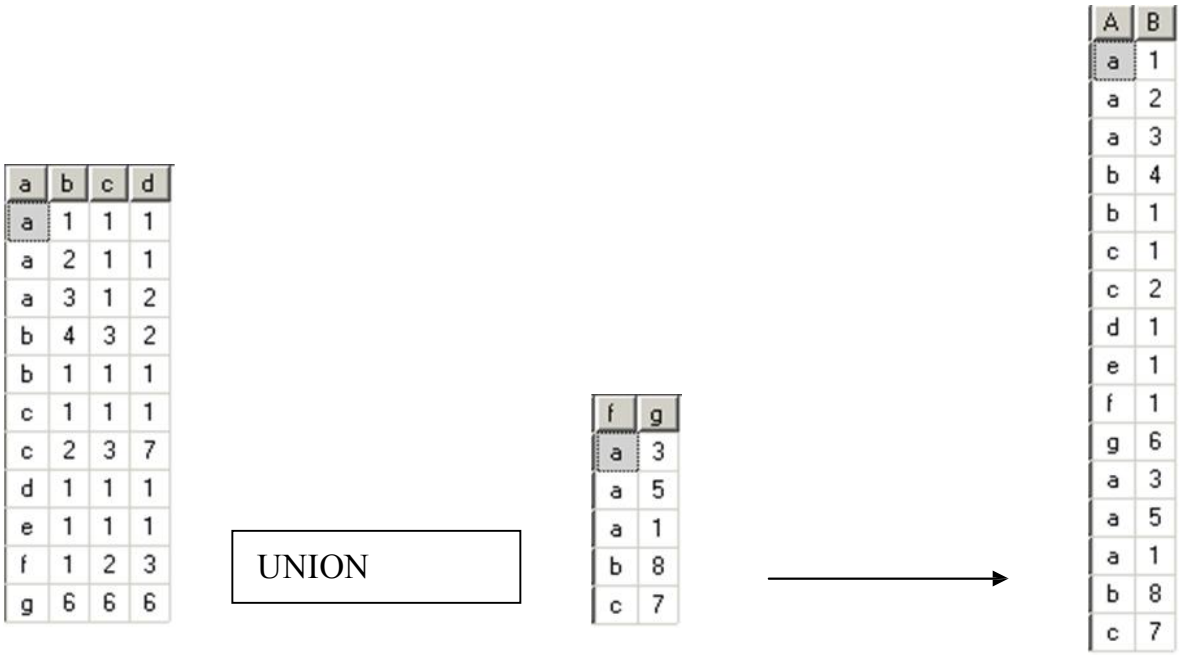


select A,B from A

union

select F,G from B

Mặc định, nếu trong các truy vấn thành phần của phép hợp xuất hiện những dòng dữ liệu giống nhau thì trong kết quả truy vấn chỉ giữ lại một dòng. Nếu muốn giữ lại các dòng này, ta phải sử dụng thêm từ khoá ALL trong truy vấn thành phần.



a	b	c	d
a	1	1	1
a	2	1	1
a	3	1	2
b	4	3	2
b	1	1	1
c	1	1	1
c	2	3	7
d	1	1	1
e	1	1	1
f	1	2	3
g	6	6	6

f	g
a	3
a	5
a	1
b	8
c	7

A	B
a	1
a	2
a	3
b	4
b	1
c	1
c	2
d	1
e	1
f	1
g	6
a	3
a	5
a	1
b	8
c	7

Khi sử dụng toán tử UNION để thực hiện phép hợp, ta cần chú ý các nguyên tắc sau:

- Danh sách cột trong các truy vấn thành phần phải có cùng số lượng.
- Các cột tương ứng trong tất cả các bảng, hoặc tập con bất kỳ các cột được sử dụng trong bản thân mỗi truy vấn thành phần phải cùng kiểu dữ liệu.
- Các cột tương ứng trong bản thân từng truy vấn thành phần của một câu lệnh UNION phải xuất hiện theo thứ tự như nhau. Nguyên nhân là do phép hợp so sánh các cột từng cột một theo thứ tự được cho trong mỗi truy vấn.
- Khi các kiểu dữ liệu khác nhau được kết hợp với nhau trong câu lệnh **UNION**, chúng sẽ được chuyển sang kiểu dữ liệu cao hơn (nếu có thể được).
- Tiêu đề cột trong kết quả của phép hợp sẽ là tiêu đề cột được chỉ định trong truy vấn đầu tiên.

Mệnh đề ORDER BY và COMPUTE dùng để sắp xếp kết quả truy vấn hoặc tính toán các giá trị thống kê chỉ được sử dụng ở cuối câu lệnh UNION. Chúng không được sử dụng ở trong bất kỳ truy vấn thành phần nào.

Mệnh đề GROUP BY và HAVING chỉ có thể được sử dụng trong bản thân từng truy vấn thành phần. Chúng không được phép sử dụng để tác động lên kết quả chung của phép hợp.

Phép toán UNION có thể được sử dụng bên trong câu lệnh INSERT.

Phép toán UNION không được sử dụng trong câu lệnh CREATE VIEW.

2.1.5 Phép nối

Khi cần thực hiện một yêu cầu truy vấn dữ liệu từ hai hay nhiều bảng, ta phải sử dụng đến phép nối. Một câu lệnh nối kết hợp các dòng dữ liệu trong các bảng khác nhau lại theo một hoặc nhiều điều kiện nào đó và hiển thị chúng trong kết quả truy vấn.

Ví dụ: Hiển thị thông tin cầu thủ có mã 7 đang thi đấu cho câu lạc bộ nào.

```
select HOTEN, VITRI, TENCLB
from CAUTHU ct, CAULACBO clb
where (ct.MACLB = Clb.MACLB)and(MACT = 7 )
```

	HOTEN	VITRI	TENCLB
1	Robinho	Tiền vệ	SHB Đà Nẵng

2.1.6 Các loại phép nối

Phép nối bằng: Một phép nối bằng (equi-join) là một phép nối trong đó giá trị của các cột được sử dụng để nối được so sánh với nhau dựa trên tiêu chuẩn bằng và tất cả các cột trong các bảng tham gia nối đều được đưa ra trong kết quả.

Một dạng đặc biệt của phép nối bằng được sử dụng nhiều là phép nối tự nhiên (natural- join). Trong phép nối tự nhiên, điều kiện nối giữa hai bảng chính là điều kiện bằng giữa khoá ngoài và khoá chính của hai bảng; Và trong danh sách chọn của câu lệnh chỉ giữ lại một cột trong hai cột tham gia vào điều kiện của phép nối.

Ví dụ : Phép kết nối bằng

```
select *
from CAUTHU ct, CAULACBO clb
where (ct.MACLB = clb.MACLB)
```

	HOTEN	VITRI	NGAYSINH	DIACHI	MACLB	MA...	S...	MACLB	TENCLB	MASAN	MATI
1	Nguyễn Vũ Phong	Tiền Đạo	1990-02-20 00:00:00.000	NULL	BBD	VN	10	BBD	Becamex Bình Dương	GD	BD
2	Nguyễn Công Vinh	Tiền Đạo	1992-03-10 00:00:00.000	NULL	HAGL	VN	9	HAGL	Hoàng Anh Gia Lai	PL	GL
3	Trần Tấn Tài	Tiền vệ	1989-11-12 00:00:00.000	NULL	BBD	VN	8	BBD	Becamex Bình Dương	GD	BD
4	Phan Hồng Sơn	Thủ môn	1991-06-10 00:00:00.000	NULL	HAGL	VN	1	HAGL	Hoàng Anh Gia Lai	PL	GL
5	Ronaldo	Tiền vệ	1989-12-12 00:00:00.000	NULL	SDN	BRA	7	SDN	SHB Đà Nẵng	CL	DN
6	Robinho	Tiền vệ	1989-10-12 00:00:00.000	NULL	SDN	BRA	8	SDN	SHB Đà Nẵng	CL	DN
7	Vidic	Hậu vệ	1987-10-15 00:00:00.000	NULL	HAGL	ANH	3	HAGL	Hoàng Anh Gia Lai	PL	GL
8	Trần Văn Santos	Thủ môn	1990-10-21 00:00:00.000	NULL	BBD	TBN	1	BBD	Becamex Bình Dương	GD	BD

Ví dụ phép kết nối tự nhiên:

```
select HOTEN, VITRI, TENCLB
from CAUTHU ct, CAULACBO clb
where (ct.MACLB = clb.MACLB)
```

	HOTEN	VITRI	TENCLB
1	Nguyễn Vũ Phong	Tiền Đạo	Becamex Bình Dương
2	Nguyễn Công Vinh	Tiền Đạo	Hoàng Anh Gia Lai
3	Trần Tấn Tài	Tiền vệ	Becamex Bình Dương
4	Phan Hồng Sơn	Thủ môn	Hoàng Anh Gia Lai
5	Ronaldo	Tiền vệ	SHB Đà Nẵng
6	Robinho	Tiền vệ	SHB Đà Nẵng
7	Vidic	Hậu vệ	Hoàng Anh Gia Lai
8	Trần Văn Santos	Thủ môn	Becamex Bình Dương
9	Nguyễn Trường Sơn	Hậu vệ	Becamex Bình Dương

Trong phép kết nối bằng, trường MACLB xuất hiện hai lần. Sự dư thừa được loại bỏ bằng cách sử dụng phép kết nối tự nhiên và việc chỉ định rõ các cột cột cần truy xuất.

Phép tự nối

Phép tự nối là phép nối mà trong đó điều kiện nối được chỉ định liên quan đến các cột của cùng một bảng. Trong trường hợp này, sẽ có sự xuất hiện tên của cùng một bảng nhiều lần trong mệnh đề FROM và do đó các bảng cần phải được đặt bí danh.

Ví dụ: Hiện thị thông tin Trân đấu, tên các câu lạc bộ tham gia thi đấu với nhau

```
select MATRAN, clb1.TENCLB as [Tên câu lạc bô 1], clb2.TENCLB as [Tên câu lạc bộ 2]
from TRANDAU td, CAULACBO clb1, CAULACBO clb2
where (td.MACLB1 = clb1.MACLB) and (td.MACLB2 = clb2.MACLB)
```

	MATRAN	Tên câu lạc bô 1	Tên câu lạc bộ 2
1	1	Becamex Bình Dương	SHB Đà Nẵng
2	2	Khatoco Khánh Hòa	Gạch Đồng Tâm Long An
3	3	SHB Đà Nẵng	Khatoco Khánh Hòa
4	4	Thép Phú Yên	Becamex Bình Dương
5	5	Thép Phú Yên	Gạch Đồng Tâm Long An
6	6	Khatoco Khánh Hòa	Becamex Bình Dương
7	8	Khatoco Khánh Hòa	Thép Phú Yên
8	9	Becamex Bình Dương	Gạch Đồng Tâm Long An

Phép nối ngoài

Trong các phép nối đã đề cập ở trên, chỉ những dòng có giá trị trong các cột được chỉ định thoả mãn điều kiện kết nối mới được hiển thị trong kết quả truy vấn, và được gọi là phép nối trong (inner join) Theo một nghĩa nào đó, những phép nối này loại bỏ thông tin chứa trong những dòng không thoả mãn điều kiện nối. Tuy nhiên, đôi khi ta cũng cần giữ lại những thông tin này bằng cách cho phép những dòng không thoả mãn điều kiện nối có mặt trong kết quả của phép nối. Để làm điều này, ta có thể sử dụng phép nối ngoài.

SQL cung cấp các loại phép nối ngoài sau đây:

- Phép nối ngoài trái (ký hiệu: *=): Phép nối này hiển thị trong kết quả truy vấn tất cả các dòng dữ liệu của bảng nằm bên trái trong điều kiện nối cho dù những dòng này không thoả mãn điều kiện của phép nối
- Phép nối ngoài phải (ký hiệu: =*): Phép nối này hiển thị trong kết quả truy vấn tất cả các dòng dữ liệu của bảng nằm bên phải trong điều kiện nối cho dù những dòng này không thoả điều kiện của phép nối.

Tuy nhiên trong SQL Server 2005 Standard Edition không hỗ trợ trực tiếp các phép nối *= và =*. Mặt khác trong các phiên bản SQL Server sắp tới các phép nối này sẽ hoàn toàn không được hỗ trợ. Do đó Microsoft khuyến cáo người sử dụng dùng các phép nối LEFT JOIN, RIGHT JOIN. Các phép nối này sẽ được nói rõ trong phần dưới đây.

2.1.7 Phép nối theo chuẩn SQL-92

Chuẩn SQL2 (SQL-92) đưa ra một cách khác để biểu diễn cho phép nối, trong cách biểu diễn này, điều kiện của phép nối không được chỉ định trong mệnh đề WHERE mà được chỉ định ngay trong mệnh đề FROM của câu lệnh. Cách sử dụng phép nối này cho phép ta biểu diễn phép nối cũng như điều kiện nối được rõ ràng, đặc biệt là trong trường hợp phép nối được thực hiện trên ba bảng trở lên.

Phép nối trong

Điều kiện để thực hiện phép nối trong được chỉ định trong mệnh đề FROM theo cú pháp như sau:

tên_bảng_1 [INNER] JOIN tên_bảng_2 ON điều_kiện_nối

Ví dụ:

Phép nối ngoài

SQL2 cung cấp các phép nối ngoài sau đây:

- Phép nối ngoài trái (LEFT OUTER JOIN)
- Phép nối ngoài phải (RIGHT OUTER JOIN)
- Phép nối ngoài đầy đủ (FULL OUTER JOIN)

Cũng tương tự như phép nối trong, điều kiện của phép nối ngoài cũng được chỉ định ngay trong mệnh đề FROM theo cú pháp:

tên_bảng_1 LEFT|RIGHT|FULL [OUTER] JOIN tên_bảng_2 ON điều_kiện_nối

Ví dụ: Hiển thị thông tin cầu thủ đá ở ví trí nào, trong câu lạc bộ nào

Thay vì viết mệnh đề WHERE như trên, ta viết các câu lệnh sau

```
select HOTEN,VITRI,TENCLB
from CAUTHU ct
inner join CAULACBO clb on(ct.MACLB = clb.MACLB)
```

Nếu phép nối ngoài trái hiển thị trong kết quả truy vấn cả những dòng dữ liệu không thoả điều kiện nối của bảng bên trái trong phép nối thì phép nối ngoài đầy đủ hiển thị trong kết quả truy vấn cả những dòng dữ liệu không thoả điều kiện nối của cả hai bảng tham gia vào phép nối.

Ví dụ: Giả sử có CSDL như sau:

MACT	HOTEN	VITRI	NGAYSINH	DIACHI	MACLB	MAQG	SO
1	Nguyễn Vũ Phong	Tiền Đạo	2/20/1990 1...	NULL	BBD	VN	10
2	Nguyễn Công Vinh	Tiền Đạo	3/10/1992 1...	NULL	HAGL	VN	9
4	Trần Tấn Tài	Tiền vệ	11/12/1989 ...	NULL	BBD	VN	8
5	Phan Hồng Sơn	Thủ môn	6/10/1991 1...	NULL	HAGL	VN	1
6	Ronaldo	Tiền vệ	12/12/1989 ...	NULL	SDN	BRA	7
7	Robinho	Tiền vệ	10/12/1989 ...	NULL	SDN	BRA	8
8	Vidic	Hậu vệ	10/15/1987 ...	NULL	HAGL	ANH	3
9	Trần Văn Santos	Thủ môn	10/21/1990 ...	NULL	BBD	TBN	1
10	Nguyễn Trường Sơn	Hậu vệ	8/26/1993 1...	NULL	BBD	VN	4
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

MACLB	TENCLB	MASAN	MATINH
BBD	Becamex Bìn...	GD	BD
GDT	Gạch Đồng ...	LA	LA
HAGL	Hoàng Anh ...	PL	GL
KKH	Khatoco Khá...	NT	KH
SDN	SHB Đà Nẵng	CL	DN
TPY	Thép Phú Yên	TH	PY
NULL	NULL	NULL	NULL

Thực hiện phép nối ngoài trái, nối ngoài phải và nối ngoài đầy đủ cho kết quả như sau:

- Phép nối ngoài trái:

```
select HOTEN,VITRI,TENCLB
from CAUTHU ct

left join CAULACBO clb on(ct.MACLB = clb.MACLB)
```

	HOTEN	VITRI	TENCLB
1	Nguyễn Vũ Phong	Tiền Đạo	Becamex Bình Dương
2	Nguyễn Công Vinh	Tiền Đạo	Hoàng Anh Gia Lai
3	Trần Tấn Tài	Tiền vệ	Becamex Bình Dương
4	Phan Hồng Sơn	Thủ môn	Hoàng Anh Gia Lai
5	Ronaldo	Tiền vệ	SHB Đà Nẵng
6	Robinho	Tiền vệ	SHB Đà Nẵng
7	Vidic	Hậu vệ	Hoàng Anh Gia Lai
8	Trần Văn Santos	Thủ môn	Becamex Bình Dương
9	Nguyễn Trường Sơn	Hậu vệ	Becamex Bình Dương

- Phép nối ngoài phải:

```
select HOTEN,VITRI,TENCLB
from CAUTHU ct

right join CAULACBO clb on(ct.MACLB = clb.MACLB)
```

	HOTEN	VITRI	TENCLB
1	Nguyễn Vũ Phong	Tiền Đạo	Becamex Bình Dương
2	Trần Tấn Tài	Tiền vệ	Becamex Bình Dương
3	Trần Văn Santos	Thủ môn	Becamex Bình Dương
4	Nguyễn Trường Sơn	Hậu vệ	Becamex Bình Dương
5	NULL	NULL	Gạch Đồng Tâm Long An
6	Nguyễn Công Vinh	Tiền Đạo	Hoàng Anh Gia Lai
7	Phan Hồng Sơn	Thủ môn	Hoàng Anh Gia Lai
8	Vidic	Hậu vệ	Hoàng Anh Gia Lai
9	NULL	NULL	Khatoco Khánh Hòa

Phép nối ngoài đầy đủ:

```
select HOTEN,VITRI,TENCLB
from CAUTHU ct

full join CAULACBO clb on(ct.MACLB = clb.MACLB)
```

	HOTEN	VITRI	TENCLB
1	Nguyễn Vũ Phong	Tiền Đạo	Becamex Bình Dương
2	Nguyễn Công Vinh	Tiền Đạo	Hoàng Anh Gia Lai
3	Trần Tấn Tài	Tiền vệ	Becamex Bình Dương
4	Phan Hồng Sơn	Thủ môn	Hoàng Anh Gia Lai
5	Ronaldo	Tiền vệ	SHB Đà Nẵng
6	Robinho	Tiền vệ	SHB Đà Nẵng
7	Vidic	Hậu vệ	Hoàng Anh Gia Lai
8	Trần Văn Santos	Thủ môn	Becamex Bình Dương
9	Nguyễn Trường Sơn	Hậu vệ	Becamex Bình Dương

Một đặc điểm nổi bật của SQL2 là cho phép biểu diễn phép nối trên nhiều bảng dữ liệu một cách rõ ràng. Thứ tự thực hiện phép nối giữa các bảng được xác định theo nghĩa kết quả của phép nối này được sử dụng trong một phép nối khác.

Ví dụ: Hiển thị các cầu thủ nước ngoài, tên câu lạc bộ tham gia

```
select HOTEN,TENCLB,TENQG
from ( CAUTHU inner join QUOCGIA on CAUTHU.MAQG = QUOCGIA.MAQG )

inner join CAULACBO on CAUTHU.MACLB = CAULACBO.MACLB
```

	HOTEN	TENCLB	TENQG
1	Nguyễn Vũ Phong	Becamex Bình Dương	Việt Nam
2	Nguyễn Công Vinh	Hoàng Anh Gia Lai	Việt Nam
3	Trần Tấn Tài	Becamex Bình Dương	Việt Nam
4	Phan Hồng Sơn	Hoàng Anh Gia Lai	Việt Nam
5	Ronaldo	SHB Đà Nẵng	Brazil
6	Robinho	SHB Đà Nẵng	Brazil
7	Vidic	Hoàng Anh Gia Lai	Anh Quốc
8	Trần Văn Santos	Becamex Bình Dương	Tây Ban Nha
9	Nguyễn Trường Sơn	Becamex Bình Dương	Việt Nam

2.1.8 Mệnh đề GROUP BY

Ngoài khả năng thực hiện các yêu cầu truy vấn dữ liệu thông thường (chiếu, chọn, nối,…) như đã đề cập như ở các phần trước, câu lệnh SELECT còn cho phép thực hiện các thao tác truy vấn và tính toán thống kê trên dữ liệu.

Mệnh đề GROUP BY sử dụng trong câu lệnh SELECT nhằm phân hoạch các dòng dữ liệu trong bảng thành các nhóm dữ liệu, và trên mỗi nhóm dữ liệu thực hiện tính toán các giá trị thống kê như tính tổng, tính giá trị trung bình,...

Các hàm gộp (aggregate functions) được sử dụng để tính giá trị thống kê cho toàn bảng hoặc trên mỗi nhóm dữ liệu. Chúng có thể được sử dụng như là các cột trong danh sách chọn của câu lệnh SELECT hoặc xuất hiện trong mệnh đề HAVING, nhưng không được phép xuất hiện trong mệnh đề WHERE

SQL cung cấp các hàm gộp dưới đây:

Hàm gộp Chức năng

SUM([ALL| DISTINCT] biểu_thức)	Tính tổng các giá trị.
AVG([ALL| DISTINCT] biểu_thức)	Tính trung bình của các giá trị
COUNT([ALL|DISTINCT] biểu_thức)	Đếm số các giá trị trong biểu thức.
COUNT(*)	Đếm số các dòng được chọn.
MAX(biểu_thức)	Tính giá trị lớn nhất MIN(biểu_thức) Tính giá trị nhỏ nhất

Hàm SUM và AVG chỉ làm việc với các biểu thức số.

Hàm SUM, AVG, COUNT, MIN và MAX bỏ qua các giá trị NULL khi tính toán.

Hàm COUNT(*) không bỏ qua các giá trị NULL.

Mặc định, các hàm gộp thực hiện tính toán thống kê trên toàn bộ dữ liệu. Trong trường hợp cần loại bỏ bớt các giá trị trùng nhau (chỉ giữ lại một giá trị), ta chỉ định thêm từ khoá DISTINCT ở trước biểu thức là đối số của hàm.

Thống kê trên toàn bộ dữ liệu

Khi cần tính toán giá trị thống kê trên toàn bộ dữ liệu, ta sử dụng các hàm gộp trong danh sách chọn của câu lệnh SELECT. Trong trường hợp này, trong danh sách chọn không được sử dụng bất kỳ một tên cột hay biểu thức nào ngoài các hàm gộp.

Ví dụ: Tính tuổi trung bình, tuổi nhỏ nhất và lớn nhất của các cầu thủ

```
Select min(year(getdate())-year(NGAYSINH)) as MINAGE, max(year(getdate())-
year(NGAYSINH)) as MAXAGE, avg(year(getdate())-year(NGAYSINH)) as AVGAGE
from CAUTHU
```

	MINAGE	MAXAGE	AVGAGE
1	18	24	21

Thống kê trên nhóm

Trong trường hợp cần thực hiện tính toán các giá trị thống kê trên các nhóm dữ liệu, ta sử dụng mệnh đề GROUP BY để phân hoạch dữ liệu vào trong các nhóm. Các hàm gộp được sử dụng sẽ thực hiện thao tác tính toán trên mỗi nhóm và cho biết giá trị thống kê theo các nhóm dữ liệu.

Ví dụ: Hiển thị số cầu thủ trong từng câu lạc bộ

```
select TENCLB, count(MACT)as [Số lượng cầu thủ]
from CAULACBO clb,CAUTHU ct
where clb.MACLB = ct.MACLB

group by TENCLB
```

	TENCLB	Số lượng cầu t...
1	Becamex Bình Dương	4
2	Hoàng Anh Gia Lai	3
3	SHB Đà Nẵng	2

Lưu ý: Trong trường hợp danh sách chọn của câu lệnh SELECT có cả các hàm gộp và những biểu thức không phải là hàm gộp thì những biểu thức này phải có mặt đầy đủ trong mệnh đề GROUP BY, nếu không câu lệnh sẽ không hợp lệ.

Mệnh đề HAVING chỉ định điều kiện trong hàm gộp

Mệnh đề HAVING được sử dụng nhằm chỉ định điều kiện đối với các giá trị thống kê được sản sinh từ các hàm gộp tương tự như cách thức mệnh đề WHERE thiết lập các điều kiện cho câu lệnh SELECT. Mệnh đề HAVING thường không thực sự có nghĩa nếu như không sử dụng kết hợp với mệnh đề GROUP BY. Một điểm khác biệt giữa HAVING và WHERE là trong điều kiện của WHERE không được có các hàm gộp trong khi HAVING lại cho phép sử dụng các hàm gộp trong điều kiện của mình.

Ví dụ: Tìm ra những câu lạc bộ có số lượng cầu thủ nước ngoài lớn hơn 2.

```
select TENCLB, count(MACT)as [Số lượng cầu thủ]
from CAULACBO clb,CAUTHU ct
where (clb.MACLB = ct.MACLB) and (ct.MAQG <> 'VN')
group by TENCLB
having count (MACT) >=2
```

	TENCLB	Số lượng cầu t...
1	SHB Đà Nẵng	2

2.1.9 Truy vấn con (Subquery)

Truy vấn con là một câu lệnh **SELECT** được lồng vào bên trong một câu lệnh **SELECT, INSERT, UPDATE, DELETE** hoặc bên trong một truy vấn con khác. Loại truy vấn này được sử dụng để biểu diễn cho những truy vấn trong đó điều kiện truy vấn dữ liệu cần phải sử dụng đến kết quả của một truy vấn khác.

Cú pháp của truy vấn con như sau:

```
(SELECT [ALL | DISTINCT] danh_sách_chọn
FROM  danh_sách_bảng
 [WHERE điều_kiện]
 [GROUP BY danh_sách_cột]
 [HAVING điều_kiện])
```

Khi sử dụng truy vấn con cần lưu ý một số quy tắc sau:

- Một truy vấn con phải được viết trong cặp dấu ngoặc. Trong hầu hết các trường hợp, một truy vấn con thường phải có kết quả là một cột (tức là chỉ có duy nhất một cột trong danh sách chọn).
- Mệnh đề COMPUTE và ORDER BY không được phép sử dụng trong truy vấn con.
- Các tên cột xuất hiện trong truy vấn con có thể là các cột của các bảng trong truy vấn ngoài.
- Một truy vấn con thường được sử dụng làm điều kiện trong mệnh đề WHERE hoặc HAVING của một truy vấn khác.
- Nếu truy vấn con trả về đúng một giá trị, nó có thể sử dụng như là một thành phần bên trong một biểu thức (chẳng hạn xuất hiện trong một phép so sánh bằng)

Phép so sánh đối với với kết quả truy vấn con

Kết quả của truy vấn con có thể được sử dụng để thực hiện phép so sánh số học với một biểu thức của truy vấn cha. Trong trường hợp này, truy vấn con được sử dụng dưới dạng:

WHERE biểu_thức phép_toán_số_học [ANY|ALL] (truy_vấn_con)

Trong đó phép toán số học có thể sử dụng bao gồm: =, <>, >, <, >=, <=; Và truy vấn con phải có kết quả bao gồm đúng một cột.

Ví dụ: Tìm câu lạc bộ có cầu thủ lớn tuổi nhất

```
select clb.TENCLB, ct.HOTEN, year(getdate()) - year(NGAYSINH) as [Tuổi]
from CAULACBO clb
inner join CAUTHU ct on clb.MACLB = ct.MACLB
where year(getdate()) - year(NGAYSINH) = (select max(year(getdate()) -
year(NGAYSINH))

from CAUTHU)
```

1	Hoàng Anh Gia Lai	Vidic	24

Nếu truy vấn con trả về nhiều hơn một giá trị, việc sử dụng phép so sánh như trên sẽ không hợp lệ. Trong trường hợp này, sau phép toán so sánh phải sử dụng thêm lượng từ ALL hoặc ANY. Lượng từ ALL được sử dụng khi cần so sánh giá trị của biểu thức với tất cả các giá trị trả về trong kết quả của truy vấn con; ngược lai, phép so sánh với lượng từ ANY có kết quả đúng khi chỉ cần một giá trị bất kỳ nào đó trong kết quả của truy vấn con thoả mãn điều kiện
Ví dụ:

Toán tử IN/NOT IN

Khi cần thực hiện phép kiểm tra giá trị của một biểu thức có xuất hiện (không xuất hiện) trong tập các giá trị của truy vấn con hay không, ta có thể sử dụng toán tử IN (NOT IN) như sau:

WHERE biểu_thức [NOT] IN (truy_vấn_con)

Ví dụ: Tìm các câu lạc bộ không tham gia thi đấu trong trận đấu nào trong mùa giải năm 2009

```
select TENCLB
from CAULACBO
where MACLB not in (SELECT MACLB1 from TRANDAU where nam = '2009')

and MACLB not in (SELECT MACLB2 from TRANDAU where nam = '2009')
```

1	Hoàng Anh Gia Lai

Truy vấn con với EXISTS

Lượng từ EXISTS được sử dụng kết hợp với truy vấn con dưới dạng:

WHERE [NOT] EXISTS (truy_vấn_con)

Lượng từ EXISTS (tương ứng NOT EXISTS) trả về giá trị True (tương ứng False) nếu kết quả của truy vấn con có ít nhất một dòng (tương ứng không có dòng nào). Điều khác biệt của việc sử dụng EXISTS với hai cách đã nêu ở trên là trong danh sách chọn của truy vấn con có thể có nhiều hơn hai cột.

Truy vấn con và mệnh đề HAVING

Một truy vấn con có thể được sử dụng trong mệnh đề HAVING của một truy vấn khác. Trong trường hợp này, kết quả của truy vấn con được sử dụng để tạo nên điều kiện đối với các hàm gộp.

Ví dụ:

2.2 Thêm, cập nhật và xóa dữ liệu

Các câu lệnh thao tác dữ liệu trong SQL không những chỉ sử dụng để truy vấn dữ liệu mà còn để thay đổi và cập nhật dữ liệu trong cơ sở dữ liệu. So với câu lệnh SELECT, việc sử dụng các câu lệnh để bổ sung, cập nhật hay xoá dữ liệu đơn giản hơn nhiều. Trong phần còn lại của chương này sẽ đề cập đến 3 câu lệnh:

Lệnh INSERT

Lệnh UPDATE

Lệnh DELETE

2.2.1 Thêm dữ liệu

Dữ liệu trong các bảng được thể hiện dưới dạng các dòng (bản ghi). Để bổ sung thêm các dòng dữ liệu vào một bảng, ta sử dụng câu lệnh INSERT.

Thêm từng dòng dữ liệu

Để bổ sung một dòng dữ liệu mới vào bảng, ta sử dụng câu lệnh INSERT với cú pháp như sau:

```
INSERT INTO tên_bảng[(danh_sách_cột)] VALUES(danh_sách_trị)
```

Ví dụ: Thêm thông tin một câu lạc bộ mới vào bảng CAULACBO

```
insert into CAULACBO
values('SLNA',N'Sông lam nghệ an','CL','KH')
```

Thêm một tập các dòng dữ liệu vào bảng

Cú pháp câu lệnh INSERT có dạng như sau:

INSERT INTO tên_bảng[(danh_sách_cột)] câu_lệnh_SELECT

Ví dụ:

```
insert into newtable
select * from CAULACBO
```

Lưu ý: Kết quả của câu lệnh SELECT phải có số cột bằng với số cột được chỉ định trong bảng đích và phải tương thích về kiểu dữ liệu.

2.2.2 Cập nhật dữ liệu

Câu lệnh UPDATE trong SQL được sử dụng để cập nhật dữ liệu trong các bảng. Câu lệnh này có cú pháp như sau:

UPDATE tên_bảng

SET tên_cột = biểu_thức

[, ..., tên_cột_k = biểu_thức_k] [FROM danh_sách_bảng] [WHERE điều_kiện]

Ví dụ:

```
update CAULACBO
set TENCLB = N'Đội tuyển SLNA'
where MACLB = 'SLNA'
```

Trong câu lệnh UPDATE có thể sử dụng CASE…WHEN.

Ví dụ:

```
update CAUTHU
set DIACHI = case when MACT <3 then N'Hà nội' else N'Nghệ An' end
```

```
(9 row(s) affected)
```

2.2.3 Xóa dữ liệu

Để xoá dữ liệu trong một bảng, ta sử dụng câu lệnh DELETE. Cú pháp của câu lệnh này như sau:

DELETE FROM tên_bảng [FROM danh_sách_bảng] [WHERE điều_kiện]

Ví dụ:

```
delete from caulacbo
where MACLB = 'SLNA'
```

Sử dụng truy vấn con trong câu lệnh DELETE

Một câu lệnh SELECT có thể được lồng vào trong mệnh đề WHERE trong câu lệnh DELETE để làm điều kiện cho câu lệnh tương tự như câu lệnh UPDATE.

Xoá toàn bộ dữ liệu trong bảng

Câu lệnh DELETE không chỉ định điều kiện đối với các dòng dữ liệu cần xoá trong mệnh đề WHERE sẽ xoá toàn bộ dữ liệu trong bảng. Thay vì sử dụng câu lệnh DELETE trong trường hợp này, ta có thể sử dụng câu lệnh TRUNCATE có cú pháp như sau:

TRUNCATE TABLE tên_bảng

3.Ngôn ngữ định nghĩa dữ liệu – DDL

Trong chương này sẽ đề cập đến nhóm các câu lệnh được sử dụng để định nghĩa và quản lý các đối tượng CSDL như bảng, khung nhìn, chỉ mục,... và được gọi là ngôn ngữ định nghĩa dữ liệu (DDL). Về cơ bản, ngôn ngữ định nghĩa dữ liệu bao gồm các lệnh:

- CREATE: định nghĩa và tạo mới đối tượng CSDL.
- ALTER: thay đổi định nghĩa của đối tượng CSDL.
- DROP: Xoá đối tượng CSDL đã có.

3.1 Tạo bảng

Câu lệnh CREATE TABLE có cú pháp như sau

CREATE TABLE tên_bảng

(

tên_cột thuộc_tính_cột các_ràng_buộc

[,...

,tên_cột_n thuộc_tính_cột_n các_ràng_buộc_cột_n]

[,các_ràng_buộc_trên_bảng]

)

Tên_bảng: tuân theo quy tắc định danh, không vượt quá 128 ký tự

Tên_cột: các cột trong bảng, mỗi bảng có ít nhất một cột.

Thuộc_tính_cột: bao gồm kiểu dữ liệu của cột, giá trị mặc định của cột, cột có được thiết lập thuộc tính *identity*, cột có chấp nhận giá trị NULL hay không. Trong đó kiểu dữ liệu là thuộc tính bắt buộc.

Các_ràng_buộc: gồm các ràng buộc về khuôn dạng dữ liệu (ràng buộc CHECK) hay các ràng buộc về bào toàn dữ liệu (PRIMARY KEY, FOREIGN KEY, UNIQUE)
Ví dụ: dưới đây tạo một bảng mới có tên CAUTHU

create table CAUTHU

(

MACT int identity (1,1) primary key,

HOTEN nvarchar(100) not null,

VITRI nvarchar(50) null,

NGAYSINH datetime null,

DIACHI nvarchar(50) null ,

MACLB varchar(5) not null,

MAQG varchar(5) not null,

SO int

)

3.2 Các loại ràng buộc

3.2.1 Ràng buộc CHECK

Ràng buộc CHECK được sử dụng nhằm chỉ định điều kiện hợp lệ đối với dữ liệu. Mỗi khi có sự thay đổi dữ liệu trên bảng (INSERT, UPDATE), những ràng buộc này sẽ được sử dụng nhằm kiểm tra xem dữ liệu mới có hợp lệ hay không.

Ràng buộc CHECK được khai báo theo cú pháp như sau:

[CONSTRAINT tên_ràng_buộc] CHECK (điều_kiện)

Ví dụ:

```
create table TRANDAU
(
MATRAN int identity(1,1) primary key,
NAM  int not null,
VONG int not null constraint chk_vong CHECK (VONG between 0 and 10),
NGAYTD datetime not null,
MACLB1 varchar(5) not null,
MACLB2 varchar(5) not null,
MASAN varchar(5) not null,
KETQUA varchar(5) not null

)
```

Thực hiện việc thêm một dòng có dữ liệu không thỏa điều kiện

```
insert into TRANDAU
values(2009,11,'2-2-2010','gh','kh','ss','6-3')
```

```
Msg 547, Level 16, State 0, Line 1
The INSERT statement conflicted with the CHECK constraint "chk_vong".
The conflict occurred in database "CSDL", table "dbo.TRANDAU", column 'VONG'.
The statement has been terminated.
```

3.2.2 Ràng buộc PRIMARY KEY

Ràng buộc PRIMARY KEY được sử dụng để định nghĩa khoá chính của bảng. Khoá chính của một bảng là một hoặc một tập nhiều cột mà giá trị của chúng là duy nhất trong bảng. Hay nói cách khác, giá trị của khoá chính sẽ giúp cho ta xác định được duy nhất một dòng (bản ghi) trong bảng dữ liệu. Mỗi một bảng chỉ có thể có duy nhất một khoá chính và bản thân khoá chính không chấp nhận giá trị NULL. Ràng buộc PRIMARY KEY là cơ sở cho việc đảm bảo tính toàn vẹn thực thể cũng như toàn vẹn tham chiếu.

Để khai báo một ràng buộc PRIMARY KEY, ta sử dụng cú pháp như sau:

[CONSTRAINT tên_ràng_buộc] PRIMARY KEY [(danh_sách_cột)]

Nếu khoá chính của bảng chỉ bao gồm đúng một cột và ràng buộc PRIMARY KEY được chỉ định ở mức cột, ta không cần thiết phải chỉ định danh sách cột sau từ khoá PRIMARY KEY. Tuy nhiên, nếu việc khai báo khoá chính được tiến hành ở mức bảng (sử dụng khi số lượng các cột tham gia vào khoá là từ hai trở lên) thì bắt buộc phải chỉ định danh sách cột ngay sau từ khóa PRIMARY KEY và tên các cột được phân cách nhau bởi dấu phẩy.

Ví dụ 1: Định nghĩa một bảng chỉ có một khóa chính

```
CREATE TABLE [dbo].[CAUTHU](
	[MACT] [numeric](18, 0) IDENTITY(1,1) PRIMARY KEY NOT NULL,

[HOTEN] [nvarchar](100) NOT NULL,
	[VITRI] [nvarchar](50) NOT NULL,
	[NGAYSINH] [datetime] NULL,
	[DIACHI] [nvarchar](200) NULL,
	[MACLB] [varchar](5) NOT NULL,
	[MAQG] [varchar](5) NOT NULL,
	[SO] [int] NOT NULL,
)
```

Hoặc là

Ví dụ 2: Định nghĩa bảng có hai khóa chính:

```
Create table HLV_CLB(
	MAHLV varchar(5) not null,
	MACLB varchar(5) not null,
	VAITRO nvarchar(100) not null,
    constraint chk_primarykey primary key (MAHLV,MACLB)
  )
```

3.2.3 Ràng buộc FOREIGN KEY

FOREIGN KEY là một cột hay một sự kết hợp của nhiều cột được sử dụng để áp đặt mối liên kết dữ liệu giữa hai table. FOREIGN KEY của một bảng sẽ giữ giá trị của PRIMARY KEY của một bảng khác và chúng ta có thể tạo ra nhiều FOREIGN KEY trong một table.

FOREIGN KEY có thể tham chiếu vào PRIMARY KEY hay cột có ràng buộc duy nhất. FOREIGN KEY có thể chứa giá trị NULL. Mặc dù mục đích chính của ràng buộc FOREIGN KEY là để kiểm soát dữ liệu chứa trong bảng có FOREIGN KEY (tức table con) nhưng thực chất nó cũng kiểm soát luôn cả dữ liệu trong bảng chứa PRIMARY KEY (tức table cha). Ví dụ nếu ta xóa dữ liệu trong bảng cha thì dữ liệu trong bảng con trở nên "mồ côi" (orphan) vì không thể tham chiếu ngược về bảng cha. Do đó ràng buộc FOREIGN KEY sẽ đảm bảo điều đó không xảy ra. Nếu bạn muốn xóa dữ liệu trong bảng cha thì trước hết bạn phải xóa hay vô hiệu hóa ràng buộc FOREIGN KEY trong bảng con trước.

Ràng buộc FOREIGN KEY được định nghĩa theo cú pháp dưới đây:

[CONSTRAINT tên_ràng_buộc] FOREIGN KEY [(danh_sách_cột)] REFERENCES tên_bảng_tham_chiếu(danh_sách_cột_tham_chiếu)
[ON DELETE CASCADE | NO ACTION | SET NULL | SET DEFAULT]
[ON UPDATE CASCADE | NO ACTION | SET NULL | SET DEFAULT]

Việc định nghĩa một ràng buộc FOREIGN KEY bao gồm các yếu tố sau:

- Tên cột hoặc danh sách cột của bảng được định nghĩa tham gia vào khoá ngoài.
- Tên của bảng được tham chiếu bởi khoá ngoài và danh sách các cột được tham chiếu đến trong bảng tham chiếu.
- Cách thức xử lý đối với các bản ghi trong bảng được định nghĩa trong trường hợp các bản ghi được tham chiếu trong bảng tham chiếu bị xoá (ON DELETE) hay cập nhật (ON UPDATE). SQL chuẩn đưa ra 4 cách xử lý

  + CASCADE: Tự động xoá (cập nhật) nếu bản ghi được tham chiếu bị xoá (cập nhật).

  + NO ACTION: (Mặc định) Nếu bản ghi trong bảng tham chiếu đang được

tham chiếu bởi một bản ghi bất kỳ trong bảng được định nghĩa thì bàn ghi đó không được phép xoá hoặc cập nhật (đối với cột được tham chiếu).

+ SET NULL: Cập nhật lại khoá ngoài của bản ghi thành giá trị NULL (nếu cột cho phép nhận giá trị NULL).

+ SET DEFAULT: Cập nhật lại khoá ngoài của bản ghi nhận giá trị mặc định (nếu cột có qui định giá trị mặc định).

3.3 Sửa đổi định nghĩa bảng

Một bảng sau khi đã được định nghĩa bằng câu lệnh CREATE TABLE có thể được sửa đổi thông qua câu lệnh ALTER TABLE. Câu lệnh này cho phép thực hiện được các thao tác sau:

- Bổ sung một cột vào bảng.
- Xoá một cột khỏi bảng.
- Thay đổi định nghĩa của một cột trong bảng.
- Xoá bỏ hoặc bổ sung các ràng buộc cho bảng

Cú pháp của câu lệnh ALTER TABLE như sau:

ALTER TABLE tên_bảng

ADD định_nghĩa_cột |

ALTER COLUMN tên_cột kiểu_dữ_liệu [NULL | NOT NULL] DROP COLUMN tên_cột |
ADD CONSTRAINT tên_ràng_buộc định_nghĩa_ràng_buộc

DROP CONSTRAINT tên_ràng_buộc

Ví dụ 1: Thêm một cột mới vào bảng ORDERS

```
alter table HLV_CLB

add description nvarchar(100) not null
```

Ví dụ 2: Thay đổi định nghĩa cột

```
alter table HLV_CLB

alter column description nvarchar(200) null
```

Ví dụ 3: Thêm ràng buộc CHECK vào cột decription

```
alter table HLV_CLB

add constraint chk_descriptionlength CHECK (len(description) > 10)
```

Ví dụ 4: Xóa ràng buộc CHECK

```
alter table HLV_CLB

drop chk_descriptionlength
```

Ví dụ 5: Xóa cột description

```
alter table HLV_CLB

drop column description
```

Ví dụ 6: Thêm một cột mới vào bảng HLV_CLB và thêm ràng buộc cho cột này

```
alter table HLV_CLB add
```

```
description nvarchar(100) null,
constraint chk_descriptionlength CHECK (len(description) > 0)
```

Nếu bổ sung thêm một cột vào bảng và trong bảng đã có ít nhất một bản ghi thì cột mới cần bổ sung phải cho phép chấp nhận giá trị NULL hoặc phải có giá trị mặc định.

Muốn xoá một cột đang được ràng buộc bởi một ràng buộc hoặc đang được tham chiếu bởi một khoá ngoài, ta phải xoá ràng buộc hoặc khoá ngoài trước sao cho trên cột không còn bất kỳ một ràng buộc và không còn được tham chiếu bởi bất kỳ khoá ngoài nào.

Nếu bổ sung thêm ràng buộc cho một bảng đã có dữ liệu và ràng buộc cần bổ sung không được thoả mãn bởi các bản ghi đã có trong bảng thì câu lệnh ALTER TABLE không thực hiện được.

3.4 Xóa bảng

Khi một bảng không còn cần thiết , ta có thể xoá nó ra khỏi cơ sở dữ liệu bằng câu lệnh DROP TABLE. Câu lệnh này cũng đồng thời xoá tất cả những ràng buộc, chỉ mục, trigger liên quan đến bảng đó.

Câu lệnh có cú pháp như sau:

DROP TABLE tên_bảng

Trong các hệ quản trị cơ sở dữ liệu, khi đã xoá một bảng bằng lệnh DROP TABLE, ta không thể khôi phục lại bảng cũng như dữ liệu của nó. Do đó, cần phải cẩn thận khi sử dụng câu lệnh này.

Câu lệnh DROP TABLE không thể thực hiện được nếu bảng cần xoá đang được tham chiếu bởi một ràng buộc FOREIGN KEY. Trong trường hợp này, ràng buộc FOREIGN KEY đang tham chiếu hoặc bảng đang tham chiếu đến bảng cần xoá phải được xoá trước.

Khi một bảng bị xoá, tất cả các ràng buộc, chỉ mục và trigger liên quan đến bảng cũng đồng thời bị xóa theo. Do đó, nếu ta tạo lại bảng thì cũng phải tạo lại các đối tượng này.

3.5 Khung nhìn - VIEW

Khung nhìn là một bảng tạm thời, có cấu trúc như một bảng, khung nhìn không lưu trữ dữ liệu mà nó được tạo ra khi sử dụng, khung nhìn là đối tượng thuộc CSDL.

Khung nhìn được tạo ra từ câu lệnh truy vấn dữ liệu (lệnh SELECT), truy vấn từ một hoặc nhiều bảng dữ liệu.

Khung nhìn được sử dụng khai thác dữ liệu như một bảng dữ liệu, chia sẻ nhiều người dùng, an toàn trong khai thác, không ảnh hưởng dữ liệu gốc.

Có thể thực hiện truy vấn dữ liệu trên cấu trúc của khung nhìn.

Như vậy, một khung nhìn trông giống như một bảng với một tên khung nhìn và là một tập bao gồm các dòng và các cột. Điểm khác biệt giữa khung nhìn và bảng là khung nhìn không được xem là một cấu trúc lưu trữ dữ liệu tồn tại trong cơ sở dữ liệu. Thực chất dữ liệu quan sát được trong khung nhìn được lấy từ các bảng thông qua câu lệnh truy vấn dữ liệu.

Câu lệnh CREATE VIEW được sử dụng để tạo ra khung nhìn và có cú pháp như sau:

CREATE VIEW tên_khung_nhìn[(danh_sách_tên_cột)] AS câu_lệnh_SELECT

Ví dụ: Tạo khung nhìn như sau

```
create view v_cauthu
as
select MACT, HOTEN, (year(getdate()) - year(NGAYSINH)) as TUOI, VITRI
from cauthu
```

Thực hiện câu truy vấn trên khung nhìn vừa tạo ra:

```
select * from v cauthu
```

	MACT	HOTEN	TU...	VITRI
1	1	Nguyễn Vũ Phong	21	Tiền Đạo
2	2	Nguyễn Công Vinh	19	Tiền Đạo
3	4	Trần Tấn Tài	22	Tiền vệ
4	5	Phan Hồng Sơn	20	Thủ môn
5	6	Ronaldo	22	Tiền vệ
6	7	Robinho	22	Tiền vệ
7	8	Vidic	24	Hậu vệ
8	9	Trần Văn Santos	21	Thủ môn
9	10	Nguyễn Trường Sơn	18	Hậu vệ

3.6 Thêm, cập nhật, xóa dữ liệu trong VIEW

Đối với một số khung nhìn, ta có thể tiến hành thực hiện các thao tác cập nhập, thêm và xoá dữ liệu. Thực chất, những thao tác này sẽ được chuyển thành những thao tác trên các bảng cơ sở và có tác động đến những bảng cơ sở.

Về mặt lý thuyết, để có thể thực hiện thao tác bổ sung, cập nhật và xoá, một khung nhìn trước tiên phải thoả mãn các điều kiện sau đây:

Trong câu lệnh SELECT định nghĩa khung nhìn không được sử dụng từ khoá DISTINCT, TOP, GROUP BY và UNION.

Các thành phần xuất hiện trong danh sách chọn của câu lệnh SELECT phải là các cột trong các bảng cơ sở. Trong danh sách chọn không được chứa các biểu thức tính toán, các hàm gộp.

Ngoài những điều kiện trên, các thao tác thay đổi đến dữ liệu thông qua khung nhìn còn phải đảm bảo thoả mãn các ràng buộc trên các bảng cơ sở, tức là vẫn đảm bảo tính toàn vẹn dữ liệu.

Mặc dù thông qua khung nhìn có thể thực hiện được thao tác bổ sung và cập nhật dữ liệu cho bảng cơ sở nhưng chỉ hạn chế đối với những khung nhìn đơn giản. Đối với những khung nhìn phức tạp thì thường không thực hiện được; hay nói cách khác là dữ liệu trong khung nhìn là chỉ đọc.

3.7 Thay đổi định nghĩa khung nhìn

Câu lệnh ALTER VIEW dùng để định nghĩa lại khung nhìn có cấu trúc như sau:

ALTER VIEW tên_khung_nhìn [(danh_sách_tên_cột)] AS

Câu_lệnh_SELECT

Ví dụ: Ví dụ dưới đây định nghĩa lại khung nhìn CUSTOMERINFO

```
alter view v_cauthu
```

```
as
select MACT, HOTEN, (year(getdate()) - year(NGAYSINH)) as TUOI, VITRI, DIACHI
from cauthu
```

Lưu ý: lệnh CREATE VIEW không làm thay đổi các quyền đã được cấp phát cho người sử dụng trước đó.

3.8 Xóa khung nhìn

Câu lệnh DROP VIEW dùng để xóa khung nhìn có cấu trúc như sau:

DROP VIEW tên_khung_nhìn

Ví dụ:

drop view v_cauthu

Lưu ý: Nếu một khung nhìn bị xoá, toàn bộ những quyền đã cấp phát cho người sử dụng trên khung nhìn cũng đồng thời bị xoá. Do đó, nếu ta tạo lại khung nhìn thì phải tiến hành cấp phát lại quyền cho người sử dụng.