NGUYỄN CÔNG TUẤN - LÊ TRẦN DIỄM QUÝ

Tự học SQL

Hãy dành 10 phút mỗi ngày bạn sẽ thành công với SQL



NHÀ XUẤT BẢN LAO ĐỘNG - XÃ HỘI

NGUYỄN CÔNG TUẤN
LÊ TRẦN DIỄM QUÝ

Tự học SQL

NHÀ XUẤT BẢN LAO ĐỘNG - XÃ HỘI

TỰ HỌC SQL

NGUYỄN CÔNG TUẤN - LÊ TRẦN DIỄM QUÝ

Chịu trách nhiệm xuất bản: NGUYỄN ĐÌNH THIÊM
Biên tập: NGUYỄN BÁ NGỌC
Sửa bản in: DIỄM QUÝ
Thiết kế bìa: LÊ VĂN TÂN

In 1.000 cuốn khổ 14,5 x 20,5cm tại công ty in Việt Hưng. Giấy đăng ký KHXB số 127-08/XB-QLXB do Cục Xuất Bản cấp ngày 07/01/2004. In xong và nộp lưu chiểu tháng 07 năm 2004.

Lời nói đầu

SQL là một công cụ quản lý dữ liệu được sử dụng phổ biến ở nhiều lĩnh vực. Nếu bạn không có nhiều thời gian để nghiên cứu sâu về tin học mà lại có nhiều dữ liệu cần phải được phân tích, xử lý, thì hãy dùng SQL vì SQL đơn giản nhưng rất hiệu quả. Hầu hết tất cả các ngôn ngữ lập trình cấp cao đều có hỗ trợ SQL. Có thể kể ra như MS Visual Basic, MS Access, MS Visual Foxpro, DBase, DB2, Paradox, Oracle...

Đầu tiên SQL được sử dụng trong các ngôn ngữ quản lý CSDL và chạy trên máy đơn lẻ. Do sự phát triển nhanh chóng của nhu cầu xây dựng những CSDL lớn theo mô hình khách/chủ (Client/Server), nhiều phần mềm sử dụng ngôn ngữ SQL đã ra đời mà điển hình là MS SQL Server, Oracle, Sybase...Trong mô hình khách/chủ, toàn bộ CSDL được tập trung lưu trữ trên máy chủ (Server), mọi thao tác xử lý dữ liệu được thực hiện trên máy chủ bằng các lệnh SQL. Máy trạm (Client) chỉ dùng để cập nhật dữ liệu cho máy chủ hoặc lấy thông tin từ máy chủ.

Trên lĩnh vực đang phát triển hiện nay là Internet, ngôn ngữ SQL lại càng đóng vai trò quan trọng hơn. Nó được sử dụng để nhanh chóng tạo ra các trang Web động (Dynamic Web Page). Trang Web động thường có nội dung được lấy ra từ CSDL. SQL có thể được sử dụng như một chất keo kết dính giữa CSDL và trang Web. Khi người dùng yêu cầu, SQL sẽ

thực hiện việc truy cập thông tin trong CSDL trên máy chủ và hiển thị kết quả trên trang Web và SQL cũng là công cụ để cập nhật thông tin cho CSDL đó.

Chính vì vậy chúng tôi biên dịch cuốn sách này nhằm đáp ứng nhu cầu tìm hiểu và là tài liệu tham khảo tốt cho các lập trình viên, sinh viên chuyên ngành công nghệ thông tin (IT) và những ai muốn tìm hiểu thêm về ngôn ngữ SQL.

Phần cuối sách là tập hợp tất cả các ví dụ, phần mềm được sử dụng trong cuốn sách này, bạn có thể tham khảo trước để có thể đọc cuốn sách được tốt hơn.

Tất cả các ví dụ, bài tập, phần mềm hỗ trợ được chúng tôi sao chép vào một đĩa CD-ROM chương trình đính kèm theo sách.

Trong quá trình thực hiện chắc chắn không tránh khỏi thiếu sót và khiếm khuyết, mong nhận được mọi ý kiến đóng góp của bạn đọc gần xa để cuốn sách ngày càng hoàn thiện hơn trong lần tái bản tiếp.

NGUYỄN CÔNG TUẤN

BÀI 1

GIỚI THIỆU TỔNG QUAN VỀ NGÔN NGỮ SQL SERVER

Mục đích chính của bài học này là giới thiệu sơ lược cho các bạn biết ngôn ngữ SQL là gì? Nó dùng để làm gì cũng như nó có thể giúp bạn được gì?

SQL là chuẩn ngôn ngữ ANSI để truy cập CSDL.

1. SQL LÀ GÌ?

SQL là viết tắt của Structured Query Language - Ngôn ngữ truy vấn cấu trúc.

- SQL cho phép bạn truy cập vào CSDL.
- SQL là một chuẩn ngôn ngữ của ANSI.
- SQL có thể thực thi các câu truy vấn trên CSDL.
- SQL có thể lấy dữ liệu từ CSDL.
- SQL có thể chèn dữ liệu mới vào CSDL.
- SQL có thể xóa dữ liệu trong CSDL.
- SQL có thể sửa đổi dữ liệu hiện có trong CSDL.

2. SQL LÀ MỘT CHUẨN:

SQL là một chuẩn của ANSI (*American National Standards Institute - Viện tiêu chuẩn quốc gia Hoa Kỳ*) về truy xuất các hệ thống CSDL. Các câu lệnh SQL được sử dụng để truy xuất và cập nhật dữ liệu trong một CSDL.

SQL hoạt động với hầu hết các chương trình CSDL như MS Access, DB2, Informix, MS SQL Server, Oracle, Sybase...

Lưu ý:

Hầu hết các chương trình CSDL hỗ trợ SQL đều có phần mở rộng cho SQL chỉ hoạt động với chính chương trình đó.

3. BẢNG CSDL:

Một CSDL thường bao gồm một hoặc nhiều bảng (table). Mỗi bảng được xác định thông qua một tên (ví dụ Customers hoặc Orders). Bảng chứa các mẩu tin hay dòng (record-row) là dữ liệu của bảng.

Dưới đây là một ví dụ về một bảng có tên là Persons (người):

LastName	FirstName	Address	City
Hansen	Ola	Timoteivn 10	Sandnes
Svendson	Tove	Borgvn 23	Sandnes
Pettersen	Kari	Storgt 20	Stavanger

Bảng ở trên bao gồm 3 mẩu tin (dòng), mỗi mẩu tin tương ứng với một người và bốn cột (LastName, FirstName, Address và City).

4. CÂU TRUY VẤN SQL:

Với SQL ta có thể truy vấn CSDL và nhận lấy kết quả trả về thông qua các câu truy vấn.

Một câu truy vấn như sau:

SELECT LastName

FROM Persons;

Sẽ trả về kết quả như sau:

LastName
Hansen
Svendson
Pettersen

Lưu ý:

SQL là ngôn ngữ thao tác dữ liệu DML (*Data Manipulation Language*). SQL là cú pháp để thực thi các câu truy vấn. SQL cũng bao gồm cú pháp để cập nhật, sửa đổi, chèn thêm và xóa các mẩu tin.

Sau đây là danh sách các lệnh và truy vấn dạng DML của SQL:

- SELECT - lấy dữ liệu từ một bảng CSDL.
- UPDATE - cập nhật/sửa đổi dữ liệu trong bảng.
- DELETE - xóa dữ liệu trong bảng.
- INSERT INTO - thêm dữ liệu mới vào bảng.

SQL là ngôn ngữ định nghĩa dữ liệu DDL (*Data Definition Language*). Phần DDL của SQL cho phép tạo ra hoặc xóa các bảng. Chúng ta cũng có thể định nghĩa các khóa (key), chỉ mục (index), chỉ định các liên kết giữa các bảng và thiết lập các quan hệ ràng buộc giữa các bảng trong CSDL.

Các lệnh DDL quan trọng nhất của SQL là:

- CREATE TABLE - tạo ra một bảng mới.
- ALTER TABLE - thay đổi cấu trúc của bảng.
- DROP TABLE - xóa một bảng.
- CREATE INDEX - tạo chỉ mục (khóa để tìm kiếm - search key).
- DROP INDEX - xóa chỉ mục đã được tạo.

Nói đến SQL bạn sẽ để cập đến các vấn đề về cơ sở dữ liệu (Database) và làm thế nào để sắp xếp, truy vấn dữ liệu một cách dễ dàng nhất đó là công việc của ngôn ngữ này. Thực chất SQL nó dựa vào cấu trúc câu lệnh chọn (Structured Query Language) từ đó nó có thể truy vấn các đối tượng trong bảng.

a) Bảng (Table):

Là nơi dùng để lưu trữ dữ liệu theo một sắp xếp nhất định mà tại đó SQL có thể truy xuất các đối tượng nằm trong bảng một cách dễ dàng. Các đối tượng trong cùng một bảng có mối liên hệ mật thiết với nhau có thể liên kết với nhau hay liên kết với các đối tượng không thuộc bảng đó. Trong bảng sẽ được thiết kế bởi các cột (Column), các dòng (Row).

b) Cột (Column):

Cột hay còn gọi là trường dữ liệu là cơ sở để tạo nên bảng, tập hợp một hay nhiều cột sẽ tạo nên bảng. Mỗi cột sẽ được định nghĩa bởi một tên để từ đó dễ dàng hiểu được đó là trường dữ liệu gì và có thể truy xuất, trong mỗi cột khác nhau có thể chứa cùng hay các kiểu dữ liệu (Datatype) khác nhau.

c) Dòng (Row):

Dòng cũng là nơi lưu trữ thông tin của bảng, mỗi dòng sẽ là một thông tin về một đối tượng (Record).

d) Khóa chính (Primary key):

Mỗi dòng sẽ được xác nhận trong bảng bởi một mã số ID cho riêng nó đó được gọi là khóa chính. Sau đây là một số đặc điểm của khóa chính:

+ Hai dòng phải có khóa chính không giống nhau.

+ Mỗi dòng đều có thể xác nhận được khóa chính, cột không thể.

+ Khóa chính sẽ không được sử dụng lại nếu như dòng đó bị xóa.

Tóm lại SQL là ngôn ngữ câu lệnh chọn rất mạnh, dễ học và phổ biến hầu hết, bạn có thể dùng SQL để thao tác trên các cơ sở dữ liệu rất lớn nhưng đạt hiệu quả nhanh nhất và chính xác cao.

5. CÁCH TẠO MỘT BẢNG TRONG SQL:

Câu lệnh:

```
CREATE TABLE <tên bảng>
(
Column1     datatype     (NULL/NOTNULL)
(CONSTRAINTS),
Column2     datatype     (NULL/NOTNULL)
(CONSTRAINTS),
                              ...
Columnn     datatype     (NULL/NOTNULL)
(CONSTRAINTS)
);
```

6. CÁCH NHẬP DỮ LIỆU VÀO BẢNG:

Câu lệnh:

```
INSERT INTO <tên bảng>
(
Column1,
Column2,
...
Columnn
)
VALUES
(
'Giá trị cột 1'
'Giá trị cột 2'
...
'Giá trị cột n'
);
```

Lưu ý:

Những trường có kiểu số sẽ không nằm trong dấu ngoặc kép ('').

BÀI 2

TRUY XUẤT DỮ LIỆU

Trong bài học này bạn sẽ tìm hiểu cách thức truy xuất một hay nhiều cột (trường) dữ liệu từ bảng (Table) thông qua câu lệnh chọn SQL.

Câu lệnh chọn **SELECT**:

Câu lệnh SELECT được dùng để truy xuất dữ liệu từ một bảng. Kết quả trả về dưới dạng bảng được lưu trong một bảng, gọi là bảng kết quả - result table (còn được gọi là tập kết quả - result set).

Kết quả trả về từ một câu truy vấn SQL được lưu trong một tập kết quả (result set). Hầu hết các hệ thống chương trình CSDL cho phép duyệt qua tập kết quả bằng các hàm lập trình như Move-To-First-Record, Get-Record-Content, Move-To-Next-Record...

Dấu chấm phẩy (;) phía sau câu lệnh:

Dấu chấm phẩy là một cách chuẩn để phân cách các câu lệnh SQL nếu như hệ thống CSDL cho phép nhiều câu lệnh SQL được thực thi thông qua một lời gọi duy nhất.

Các câu lệnh SQL trong bài viết này đều là các câu lệnh đơn (mỗi câu lệnh là một và chỉ một lệnh SQL). MS Access và MS SQL Server không đòi hỏi phải có dấu chấm phẩy ngay sau mỗi câu lệnh SQL, nhưng một số chương trình CSDL khác có thể bắt buộc bạn phải thêm dấu chấm phẩy sau mỗi câu lệnh SQL (cho dù đó là câu lệnh đơn).

1. TRUY XUẤT MỘT CỘT:

Ví dụ:

Bạn cần truy xuất cột tên sản phẩm có trong bảng sản phẩm thì cú pháp:

SELECT <cột i>
FROM <tên bảng>;
i=(1,n).

Giả sử bạn cần lấy các thông tin về cột mã số sản phẩm (pro_id) từ bảng sản phẩm (products) chẳng hạn thì làm như sau :

Nhập vào:

SELECT PRO_ID
FROM PRODUCTS;

Lưu ý:

Bạn hãy nhấn phím F5 để thi hành lệnh.

Màn hình xuất ra:

	pro_id
1	BR02
2	BR01
3	BR03
4	BNBG01
5	BNBG02
6	BNBG03
7	RGAN01
8	RYL01
9	RYL02

2. TRUY XUẤT NHIỀU CỘT:

Giả sử bạn cần lấy các thông tin về cột mã số sản phẩm (pro_id), mã số đại lí (vend_id) và tên các sản phẩm (pro_name) chẳng hạn thì làm như sau :

Cú pháp:

SELECT <cột1,cột2,...>
FROM <tên bảng>;

Nhập vào:

SELECT pro_id,vend_id,pro_name
FROM products;

Màn hình xuất ra:

	pro_id	vend_id	pro_name
1	BR02	BRS01	12 inch teddy bear
2	BR01	BRS01	8 inch teddy bear
3	BR03	BRS01	18 inch teddy bear
4	BNBG01	DLL01	Fish bean bag toy
5	BNBG02	DLL01	Bird bean bag toy
6	BNBG03	DLL01	Rabbit bean bag toy
7	RGAN01	DLL01	Raggedy Ann
8	RYL01	FNG01	King Doll
9	RYL02	FNG01	Queen Doll

3. TRUY XUẤT TẤT CẢ CÁC CỘT:

Ví dụ: Bạn cần truy xuất tất cả các cột có trong bảng sản phẩm thì cú pháp:

Nhập vào:

SELECT *
FROM SẢN PHẨM;

Dấu (*) mô tả cho bạn về sự lựa chọn tất cả các cột.

Màn hình xuất ra:

	pro_id	vend_id	pro_name	pro_price	pro_decs
1	BR02	BRS01	12 inch teddy bear	8.99	12 inch teddy bear,come ...
2	BR01	BRS01	8 inch teddy bear	5.99	8 inch teddy bear,come w ...
3	BR03	BRS01	18 inch teddy bear	11.99	18 inch teddy bear,come ...
4	BNBG01	DLL01	Fish bean bag toy	3.49	Fish bean bag toy, comple...
5	BNBG02	DLL01	Bird bean bag toy	3.49	Bird bean bag toy,eggs a...
6	BNBG03	DLL01	Rabbit bean bag toy	3.49	Rabbit bean bag toy, com
7	RGAN01	DLL01	Raggedy Ann	4.99	18 inch Raggedy Ann doll
8	RYL01	FNG01	King Doll	9.49	12 inch king doll with royal...
9	RYL02	FNG01	Queen Doll	9.49	12 inch king doll with royal...

BÀI 3

SẮP XẾP DỮ LIỆU

Mục đích chính của bài này là sử dụng câu lệnh chọn (SELECT) kết hợp với sắp xếp lại dữ liệu (ORDER BY) theo một trật tự được qui định.

1. SẮP XẾP MỘT CỘT:

Ví dụ: bạn cần sắp xếp các tên sản phẩm trong cột tên sản phẩm của bảng sản phẩm thì cú pháp như sau:

Nhập vào:

```
SELECT pro_name
FROM products
ORDER BY pro_name;
```

Màn hình xuất ra:

	pro_name
1	12 inch teddy bear
2	18 inch teddy bear
3	8 inch teddy bear
4	Bird bean bag toy
5	Fish bean bag toy
6	King Doll
7	Queen Doll
8	Rabbit bean bag toy
9	Raggedy Ann

Một lưu ý là mệnh đề sắp xếp ORDER BY phải được đặt tại điểm cuối của dòng lệnh. Tương tự như trên bạn cũng có thể tiến hành sắp xếp nhiều cột cùng lúc:

Nhập vào:

```
SELECT pro_id,vend_id,pro_name
FROM products
ORDER BY pro_id,vend_id,pro_name;
```

Màn hình xuất ra:

	pro_id	vend_id	pro_name
1	BNBG01	DLL01	Fish bean bag toy
2	BNBG02	DLL01	Bird bean bag toy
3	BNBG03	DLL01	Rabbit bean bag toy
4	BR01	BRS01	8 inch teddy bear
5	BR02	BRS01	12 inch teddy bear
6	BR03	BRS01	18 inch teddy bear
7	RGAN01	DLL01	Raggedy Ann
8	RYL01	FNG01	King Doll
9	RYL02	FNG01	Queen Doll

2. SẮP XẾP THEO THỨ TỰ CỘT:

Ngoài chức năng sắp xếp theo tên cột SQL còn hỗ trợ chức năng sắp xếp theo vị trí cột đơn giản hơn, cú pháp:

```
SELECT pro_id,vend_id,pro_name
FROM products
ORDER BY 1,2,3;
```

Màn hình xuất ra:

	pro_id	vend_id	pro_name
1	BNBG01	DLL01	Fish bean bag toy
2	BNBG02	DLL01	Bird bean bag toy
3	BNBG03	DLL01	Rabbit bean bag toy
4	BR01	BRS01	8 inch teddy bear
5	BR02	BRS01	12 inch teddy bear
6	BR03	BRS01	18 inch teddy bear
7	RGAN01	DLL01	Raggedy Ann
8	RYL01	FNG01	King Doll
9	RYL02	FNG01	Queen Doll

Trong đó 1, 2, 3 sẽ là các vị trí thay thế cho các cột tương ứng pro_id,vend_id và pro_name.

Mệnh đề **order by** trên nó sẽ mặc định cho bạn sắp xếp theo thứ tự từ A đến Z (tăng dần), nếu bạn muốn sắp xếp theo thứ tự giảm dần bạn phải thêm **desc** sau mệnh đề **order by**, cú pháp:

```
SELECT pro_id,pro_price,pro_name
FROM products
ORDER BY 2 desc;
```

Màn hình xuất ra:

	pro_id	pro_price	pro_name
1	BR03	11.99	18 inch teddy bear
2	RYL01	9.49	King Doll
3	RYL02	9.49	Queen Doll
4	BR02	8.99	12 inch teddy bear
5	BR01	5.99	8 inch teddy bear
6	RGAN01	4.99	Raggedy Ann
7	BNBG02	3.49	Bird bean bag toy
8	BNBG03	3.49	Rabbit bean bag toy
9	BNBG01	3.49	Fish bean bag toy

Nếu bạn muốn sort nhiều cột cùng lúc nhưng chỉ có cột pro_price là sort theo thứ tự giảm dần thì:

```
SELECT pro_id,pro_price,pro_name
FROM products
ORDER BY 2 desc,3;
```

Màn hình xuất ra:

BR03	11.99	18 inch teddy bear
RYL01	9.49	King Doll
RYL02	9.49	Queen Doll
BR02	8.99	12 inch teddy bear
BR01	5.99	8 inch teddy bear
RGAN01	4.99	Raggedy Ann
BNBG02	3.49	Bird bean bag toy
BNBG01	3.49	Fish bean bag toy
BNBG03	3.49	Rabbit bean bag toy

BÀI 4

LỌC DỮ LIỆU

Trong bài học này bạn sẽ học được cách dùng câu lệnh chọn với mệnh đề where để lọc dữ liệu theo một điều kiện nào đó.

Ví dụ:

```
select pro_name,pro_price
from products
where pro_price=3.49;
```

Màn hình xuất ra:

	pro_name	pro_price
1	Fish bean bag toy	3.49
2	Bird bean bag toy	3.49
3	Rabbit bean bag toy	3.49

Trong trường hợp này đối tượng bạn cần lọc là 3.49 ở cột pro_price. Bạn nên lưu ý mệnh đề **where** phải nằm trước mệnh đề **order by** (sort, nếu có sử dụng).

Ngoài ra, trong SQL bạn còn có khả năng sử dụng các toán tử (equality) để lọc dữ liệu theo miền mong muốn. Bảng các toán tử được sử dụng:

TOÁN TỬ	DIỄN GIẢI
=	ngang bằng
<>	không ngang bằng
!=	không ngang bằng
<	nhỏ hơn
<=	nhỏ hơn hoặc bằng
!<	không nhỏ hơn
>	lớn hơn
>=	lớn hơn hoặc bằng
!>	không lớn hơn
BETWEEN	giữa hai giá trị
IS NULL	không có giá trị

Bây giờ bạn cần lọc trong bảng ra các giá trị có đơn giá sản phẩm nhỏ hơn hoặc bằng 8:

```
select pro_name,pro_price
from products
where pro_price<=8
order by pro_price;
```

Màn hình xuất ra:

	pro_name	pro_price
1	8 inch teddy bear	5.99
2	Raggedy Ann	4.99
3	Fish bean bag toy	3.49
4	Bird bean bag toy	3.49
5	Rabbit bean bag toy	3.49

Tương tự ta cũng có thể dùng các toán tử khác để truy vấn các đối tượng cần tìm.

Ví dụ ta cần tìm ra các đối tượng có mã số đại lí không phải là "DLL01", cú pháp:

```
select vend_id,pro_name
from products
where vend_id<>'dll01'
order by pro_name;
```

Màn hình xuất ra:

	vend_id	pro_name
1	BRS01	12 inch teddy bear
2	BRS01	18 inch teddy bear
3	BRS01	8 inch teddy bear
4	FNG01	King Doll
5	FNG01	Queen Doll

Hay bạn cần tìm các giá trị trong bảng products các đơn giá nằm giữa 5 và 10 thì cú pháp:

```
select pro_name,pro_price
from products
where pro_price between 5 and 10
order by pro_price;
```

Màn hình xuất ra:

	pro_name	pro_price
1	8 inch teddy bear	5.99
2	12 inch teddy bear	8.99
3	King Doll	9.49
4	Queen Doll	9.49

Hay bạn cũng có thể tìm trong bảng products ở cột pro_price các đối tượng không mang giá trị, cú pháp:

```
select pro_name
from products
where pro_price is null
order by pro_name desc;
```

Sẽ cho kết quả là không có giá trị nào cả, bởi vì trong bảng products của chúng ta tất cả các giá đều mang giá trị (not null).

Lưu ý:

NULL mang ý nghĩa là đối tượng đó có giá trị bằng 0 hay không có gì cả.

Tóm lại bạn có thể sử dụng tất cả các toán tử khác tùy theo nhu cầu để lọc hay truy vấn các giá trị bạn cần tìm.

BÀI 5

TRÌNH LỌC DỮ LIỆU CAO CẤP

Trong bài học này bạn sẽ được học cách nối mệnh đề where với các điều kiện để tạo ra sự lọc một cách thông minh. Bạn cũng sẽ được tìm hiểu về các toán tử Not, In.

Các toán tử logic And, Or, cú pháp:

```
select pro_id,pro_price,pro_name
from products
where vend_id='DLL01' and pro_price<=4;
```

Màn hình xuất ra:

	pro_id	pro_price	pro_name
1	BNBG01	3.49	Fish bean bag toy
2	BNBG02	3.49	Bird bean bag toy
3	BNBG03	3.49	Rabbit bean bag toy

Ví dụ trên truy xuất cho bạn tất cả các đối tượng có vend_id là DDL01 và giá nhỏ hơn hoặc bằng 4, có nghĩa là các đối tượng trên phải đồng thời thỏa mãn hai điều kiện được đặt ra.

```
select pro_price,pro_name
from products
where vend_id='DLL01' or vend_id='BRS01';
```

Màn hình xuất ra:

	pro_name	pro_price
1	12 inch teddy bear	8.99
2	8 inch teddy bear	5.99
3	18 inch teddy bear	11.99
4	Fish bean bag toy	3.49
5	Bird bean bag toy	3.49
6	Rabbit bean bag toy	3.49
7	Raggedy Ann	4.99

Kết hợp giữa hai toán tử And và Or:

```
select pro_price,pro_name
from products
where (vend_id='DLL01' or vend_id='BRS01') And pro_price>=10;
```

Màn hình xuất ra:

	pro_price	pro_name
1	11.99	18 inch teddy bear

Bạn cũng có thể kết hợp theo một cách khác, cú pháp:

```
select pro_name,pro_price
from products
where vend_id='DLL01' or vend_id=BRS01 And prod_price>=10;
```

Màn hình xuất ra:

	pro_name	pro_price
1	18 inch teddy bear	11.99
2	Fish bean bag toy	3.49
3	Bird bean bag toy	3.49
4	Rabbit bean bag toy	3.49
5	Raggedy Ann	4.99

Qua hai ví dụ trên bạn sẽ phân biệt được sự khác nhau trong việc các điều kiện khác nhau sẽ qui định nên sự xuất ra màn hình của các đối tượng.

Sử dụng toán tử **IN**: có thể hiểu một cách ngắn gọn toán tử **IN** được sử dụng như toán tử **OR** song ngắn gọn hơn.

```
select pro_name,pro_price
from products
where vend_id IN ('DLL01','BRS01')
order by pro_name;
```

Hoặc:

```
select pro_name,pro_price
from products
where vend_id='DLL01' OR vend_id='BRS01'
order by pro_name;
```

Màn hình xuất ra:

	pro_name	pro_price
1	12 inch teddy bear	8.99
2	18 inch teddy bear	11.99
3	8 inch teddy bear	5.99
4	Bird bean bag toy	3.49
5	Fish bean bag toy	3.49
6	Rabbit bean bag toy	3.49
7	Raggedy Ann	4.99

Nhìn chung toán tử **IN** có thể được dùng để thay thế **OR**, nhằm tránh sự rườm rà trong cú pháp câu lệnh khi làm việc với một số lượng lớn các đối tượng.

Sử dụng toán tử **NOT**: là một mệnh đề phủ định, thường

được sử dụng đi kèm với các toán tử khác, không sử dụng độc lập. Điều đặc biệt là toán tử **NOT** được dùng trước các cột để lọc nó. Có thể nói **NOT** và <> trên nguyên tắc rất giống nhau về ý nghĩa, song **NOT** tốt hơn nhiều và ít rắc rối hơn khi ta làm việc trên các mảng dữ liệu lớn, cú pháp:

```
select pro_name
from products
where NOT vend_id='DLL01'
order by pro_name;
```

Hoặc:

```
select pro_name
from products
where vend_id<>'DLL01'
order by pro_name;
```

Màn hình xuất ra:

	pro_name
1	12 inch teddy bear
2	18 inch teddy bear
3	8 inch teddy bear
4	King Doll
5	Queen Doll

đại diện chặn tại điểm đầu và điểm cuối của cụm từ đó.

Ví dụ:

```
select pro_id,pro_name
from products
where pro_name like '%bean bag%';
```

Màn hình xuất ra:

	vend_id	pro_name	pro_id
1	DLL01	Fish bean bag toy	BNBG01
2	DLL01	Bird bean bag toy	BNBG02
3	DLL01	Rabbit bean bag toy	BNBG03

Thêm vào đó bạn cũng có thể truy xuất các đối tượng mà ta chỉ biết kí tự đầu tiên và kí tự cuối cùng, khi đó ta đặt kí tự đại diện % ở khoảng giữa hai kí tự.

Ví dụ:

```
select vend_id,pro_name,pro_id
from products
where pro_name like 'k%l';
```

Màn hình xuất ra:

	vend_id	pro_name	pro_id
1	FNG01	King Doll	RYL01

Kí tự đại diện % cũng có thể đại biểu cho zero, một hay nhiều kí tự phụ thuộc vào cách đặt đặc biệt của %.

2. KÍ TỰ ĐẠI DIỆN (_):

Trên nguyên tắc kí tự đại diện (_) được sử dụng giống như (%) song có một điều khác biệt là (_) chỉ đại biểu cho duy nhất một kí tự.

Ví dụ:

```
select vend_id,pro_name,pro_id
from products
where pro_name like '_ inch teddy bear';
```

Màn hình xuất ra:

	vend_id	pro_name	pro_id
1	BRS01	8 inch teddy bear	BR01

Hoặc:

```
select vend_id,pro_name,pro_id
from products
where pro_name like '__ inch teddy bear';
```

Màn hình xuất ra:

	vend_id	pro_name	pro_id
1	BRS01	12 inch teddy bear	BR02
2	BRS01	18 inch teddy bear	BR03

Theo trên nếu bạn truy xuất một kí tự đại biểu (_) thì kết quả là 8 inch teddy bear vì 8 chỉ có một kí tự còn nếu bạn truy xuất theo hai kí tự đại biểu (_ _) thì kết quả sẽ là 12 inch teddy bear và 18 inch teddy bear.

Còn nếu bạn dùng % thì nó sẽ đại biểu cho không, một, hai,... kí tự trong khi đó (_) chỉ đại biểu cho một và chỉ một kí tự mà thôi, đây chính là điểm khác biệt cơ bản nhất giữa % và (_).

Ví dụ:

```
select vend_id,pro_name,pro_id
from products
where pro_name like '% inch teddy bear';
```

Màn hình xuất ra:

	vend_id	pro_name	pro_id
1	BRS01	12 inch teddy bear	BR02
2	BRS01	8 inch teddy bear	BR01
3	BRS01	18 inch teddy bear	BR03

3. KÍ TỰ ĐẠI DIỆN KHOẢNG TRỐNG ([ ]):

Chức năng chính của kí tự đại biểu ([ ]) là dò tìm ra các đối tượng với một đặc điểm mà đối tượng đó mang.

Ví dụ:

```
select cust_contact
from customerstable
where cust_contact like '[jm]%'
order by cust_contact;
```

Màn hình xuất ra:

	cust_contact
1	jim jones
2	john smith
3	michelle green

Trong ví dụ trên sẽ truy xuất cho các bạn những người có tên với từ đầu tiên là các chữ J và M trong cột cust_contact.

Nếu như bạn cần loại tất cả những người có tên bắt đầu bằng J và M ra khỏi danh sách truy tìm tức là ngược với ví dụ trên thì cú pháp:

```
select cust_contact
from customerstable
where cust_contact like '[^jm]%'
order by cust_contact;
```

Màn hình xuất ra:

	cust_contact
1	Denise l. stephens
2	kim howard

Lẽ dĩ nhiên bạn cũng có thể sử dụng một cú pháp với toán tử **NOT** để cho ra một kết quả tương tự:

```
select cust_contact
from customerstable
where NOT cust_contact like '[jm]%'
order by cust_contact;
```

Một điều cần lưu ý là bạn đừng nên quá lạm dụng vào các kí tự đại diện, chỉ có thể sử dụng khi thật sự cần thiết.

BÀI 7

TẠO CẤU TRÚC CỘT DỮ LIỆU

Trong bài học này bạn sẽ học thế nào là cấu trúc cột dữ liệu, cách tạo và liên kết chúng với nhau.

Cấu trúc cột dữ liệu là thuật ngữ dùng để chỉ các cách thức định dạng các cột, liên kết cột.

Ví dụ trong bảng customerstable bạn cần liên kết 2 cột vend_state, vend_zip lại với nhau, cú pháp:

```
SELECT VEND_CITY+','+VEND_STATE+' '+VEND_ZIP
FROM VENDORS
ORDER BY VEND_NAME;
```

Nếu bạn đang sử dụng phiên bản Oracle thì cú pháp:

```
SELECT VEND_CITY||','||VEND_STATE||' '||VEND_ZIP
FROM VENDORS
ORDER BY VEND_NAME;
```

Màn hình xuất ra:

	(No column name)
1	ANY TOWN ,OH 44333
2	BEAR TOWN ,MI 44444
3	DOLLSVILLE ,CA 99999
4	LONDON , N16 6PS
5	NEWYORK ,NP 11111
6	PARIS , 45678

Lúc này hai cột vend_state, vend_zip sẽ được dồn về cột vend_city.

Trong ví dụ trên mặc dù dữ liệu đã được dồn vào một cột (vend_city) song dữ liệu được lưu dưới dạng text, còn tồn tại nhiều khoảng trống (spaces), bạn sẽ phải định dạng nó với RTRIM sẽ xóa hết các khoảng trắng đó, cú pháp:

```
select rtrim (vend_city)+','+rtrim(vend_state)+''+rtrim(vend_zip)
from vendors
order by vend_name;
```

Nếu bạn đang sử dụng phiên bản **Oracle** thì cú pháp:

```
select rtrim (vend_city)||','||rtrim(vend_state)||''||rtrim(vend_zip)
from vendors
order by vend_name;
```

Màn hình xuất ra:

	(No column name)
1	ANY TOWN,OH44333
2	BEAR TOWN,MI44444
3	DOLLSVILLE,CA99999
4	LONDON,N16 6PS
5	NEWYORK,NP11111
6	PARIS,45678

Bạn cần lưu ý thêm một số thông tin sau: LTRIM() sẽ cắt các khoảng trắng từ bên trái chuỗi, TRIM() sẽ cắt khoảng trắng từ hai phía, RTRIM() sẽ cắt khoảng trắng từ bên phải chuỗi. Song một vấn đề được đặt ra là sau khi liên kết các cột lại bạn sẽ thấy một cột không có tên (no name column), vấn đề là bạn phải đặt tên (alias) cho nó, cú pháp:

```
select rtrim (vend_city)+','+rtrim(vend_state)+''+rtrim(vend_zip)
AS NGUYENCONGTUAN
from vendors
order by vend_name;
```

Nếu bạn đang sử dụng phiên bản Oracle thì cú pháp:

```
select rtrim (vend_city)||','||rtrim(vend_state)||"||rtrim(vend_zip)
AS NGUYENCONGTUAN
from vendors
order by vend_name;
```

Màn hình xuất ra:

	NGUYENCONGTUAN
1	ANY TOWN,OH44333
2	BEAR TOWN,MI44444
3	DOLLSVILLE,CA99999
4	LONDON,N16 6PS
5	NEWYORK,NP11111
6	PARIS,45678

Kết thúc mục này bạn sẽ tiến hành một truy xuất kết hợp với calculated fields, cú pháp:

```
select vend_name,vend_address,
rtrim (vend_city)+','+rtrim(vend_state)+"+rtrim(vend_zip)
AS NGUYENCONGTUAN
from vendors
order by vend_name;
```

Hoặc đối với phiên bản Oracle thì cú pháp:

```
select vend_name,vend_address,
rtrim (vend_city)||','||rtrim(vend_state)||"||rtrim(vend_zip) AS
NGUYENCONGTUAN
from vendors
order by vend_name;
```

Màn hình xuất ra:

	vend_name	vend_address	NGUYENCONGTUAN
1	BEAR EMPORIUM	500 PARK STREET	ANY TOWN,OH44333
2	BEARS RUS	123 MAIN STREET	BEAR TOWN,MI44444
3	DOLL HOUSE INC	555 HIGH STREET	DOLLSVILLE,CA99999
4	FUN AND GAMES	42 GALAXY ROAD	LONDON,N16 6PS
5	FURBALL INC	1000 5TH AVENUE	NEWYORK,NP11111
6	JOUETS ET OURS	1 RUE AMUSEMENT	PARIS,45678

Ngoài ra calculated fields còn thực hiện được các phép toán, cú pháp:

```
select *
from orderitems
where order_num=20008;
```

Màn hình xuất ra:

	order_num	oder_item	prod_id	quantity	tem_price
1	20008	1	rgan01	5	4.99
2	20008	2	br03	5	11.99
3	20008	3	bnbg01	10	3.49
4	20008	4	bnbg02	10	3.49
5	20008	5	bnbg03	10	3.49

Lúc này nó sẽ truy xuất cho bạn tất cả các đối tượng có order_num là 20008.

Hay bạn có thể kết hợp các phép toán với việc thêm cột trong bảng, cú pháp:

```
select prod_id,
       quantity,
       item_price,
       quantity*item_price as expanded_price
from orderitems
where order_num=20008;
```

Màn hình xuất ra:

	ɔrod_id	ɥuantity	tem_price	expanded_price
1	rgan01	5	4.99	24.95
2	br03	5	11.99	59.95
3	bnbg01	10	3.49	34.90
4	bnbg02	10	3.49	34.90
5	bnbg03	10	3.49	34.90

Các phép tính được dùng trong câu lệnh:

Toán tử	Diễn giải
+	cộng
-	trừ
*	nhân
/	chia

BÀI 8

SỬ DỤNG HÀM DỮ LIỆU

Trong bài học này bạn sẽ tìm hiểu thế nào là hàm dữ liệu? các kiểu hàm? Cách sử dụng các hàm?

Cũng giống như mọi ngôn ngữ khác SQL server cũng cung cấp các hàm điều khiển, song không giống như các câu lệnh (statements), các hàm của SQL server không thể làm việc với nhiều hệ thống khác nhau (not portable). Sau đây bạn sẽ tìm hiểu các hàm về: văn bản, số, ngày giờ và hệ thống.

HÀM DỮ LIỆU TEXT:

1. HÀM UPPER:

Cú pháp:

```
select vend_name,upper(vend_name) as vend_name_upcase
from vendors
order by vend_name;
```

Màn hình xuất ra:

	vend_name	vend_name_upcase
1	Bears Emporium	BEARS EMPORIUM
2	Bears R Us	BEARS R US
3	Doll House Inc.	DOLL HOUSE INC.
4	Fun and Games	FUN AND GAMES
5	FURBALL INC.	FURBALL INC.
6	Jouets et ours	JOUETS ET OURS

Trong ví dụ trên hàm upper sẽ nở hoa tất cả các phần tử nằm trong trường vend_name và tách ra thành một trường mới mang tên vend_name_upcase.

Sau đây là một số hàm chuẩn dạng text:

Hàm	Diễn Giải
LEFT()	Trả về các kí tự phía bên trái chuỗi.
LENGTH(), DATALENGHT(), LEN()	Cho ra độ dài chuỗi.
LOWER()	Chuyển chuỗi ra chữ thường.
LTRIM()	Cắt các khoảng trắng ở bên trái chuỗi
RIGHT()	Trả về các kí tự phía bên phải chuỗi.
RTRIM()	Cắt các khoảng trắng ở bên phải chuỗi.
SOUNDEX()	Trả về chuỗi với giá trị hàm Sounddex
UPPER()	Chuyển chuỗi ra chữ hoa.

2. HÀM SOUNDEX

Soundex là hàm có tác dụng dò tìm cho bạn đối tượng thuộc trường gần giống, cú pháp:

Ví dụ do lỗi bàn phím bạn gõ nhầm 'Michelle' thành 'Michalle' chẳng hạn thì kết quả sẽ là chẳng có gì cả:

```
select cust_name,cust_contact
from customers
where cust_contact='Michalle Green';
```

Màn hình xuất ra:

	cust_name	cust_contact	

Nhưng nếu bạn nhờ hàm soundex() dò tìm giúp bạn thì:

```
select cust_name,cust_contact
from customers
where soundex(cust_contact)=soundex('Michalle Green');
```

Màn hình xuất ra:

	cust_name	cust_contact
1	Kids Place	Michelle Green

3. HÀM NGÀY VÀ THỜI GIAN:

Giả sử bạn cần truy xuất tất cả các hóa đơn (orders) vào năm 2001 thì cú pháp như sau:

```
select order_num
from orders
where datepart(yy,order_date)=2001
```

Màn hình xuất ra:

	order_num
1	20005
2	20006
3	20007
4	20008
5	20009

Song nếu bạn đang sử dụng ngôn ngữ **Oracle** thì có nhiều cú pháp:

```
select order_num
from orders
where to_number(to_char(order_date,'yy'))=2001;
```

Hoặc:

```
select order_num
from orders
where order_date between to_date('01-jan-2001') and
to_date('31-dec-2001');
```

Nếu bạn muốn truy xuất các hóa đơn vào February 2001 thì thực hiện như sau:

```
select order_num
from orders
where datepart(yy,order_date)=2001 and
datepart(mm,order_date)=2;
```

	order_num
1	20008
2	20009

Nếu bạn đang sử dụng ngôn ngữ **Oracle** thì có các cú pháp sau:

```
select order_num
from orders
where to_number(to_char(order_date,'yy'))=2001 and
to_number(to_char(order_date,'mm'))=2;
```

Hoặc:

```
select order_num
from orders
where order_date between to_date('01-feb-2001') and
to_date('28-feb-2001');
```

4. HÀM SỐ:

Sau đây là bảng một số hàm số thông dụng nhất

HÀM	DIỄN GIẢI
ABS()	Trả về giá trị tuyệt đối
COS()	Tính hàm cosin
EXP()	Tính hàm mũ
PI()	Trả về số Pi
SIN()	Tính hàm sin
SQRT()	Tính căn bậc hai
TAN()	Tính hàm tang

BÀI 9

TỔNG DỮ LIỆU

Trong bài học này bạn sẽ tìm hiểu các hàm tính tổng cũng như cách sử dụng chúng để tổng các dữ liệu có trong bảng.

Sau đây là bảng một số hàm tính tổng thông dụng nhất của SQL server:

HÀM	DIỄN GIẢI
AVG()	Trả về giá trị trung bình của cột
COUNT()	Đếm số dòng của cột
MAX()	Đưa ra phần tử cao nhất cột
MIN()	Đưa ra phần tử thấp nhất cột
SUM()	Tính tổng các phần tử trong cột

1. HÀM TRUNG BÌNH:

Giả sử bạn cần tìm giá trị trung bình của cột giá sản phẩm (pro_price) có trong bảng products thì cú pháp như sau:

```
select avg(pro_price) as avg_price
from products;
```

Màn hình xuất ra:

	avg_price
1	6.823333

Ngoài ra AVG() còn có khả năng tính trung bình của cột đặc biệt:

```
select avg(pro_price) as avg_price
from products
where vend_id='DLL01';
```

Màn hình xuất ra:

	avg_price
1	3.865000

Trong ví dụ trên sẽ xuất ra giá trung bình của các giá sản phẩm có mã số đại lí (vend_id) là DLL01.

2. HÀM COUNT():

Hàm count() sẽ đếm xem trong bảng chỉ định của bạn có bao nhiêu dòng (Records), nếu:

- Count(*): sẽ cho ra tất cả các dòng kể cả các cột không bao gồm giá trị (null column).
- Count(column): cho ra các cột có giá trị và loại ra cột không mang giá trị.

Ví dụ:

```
select count(*) as num_cust
from customers;
```

Màn hình xuất ra:

	num_cust
1	5

Hoặc:

```
select count(cust_email) as num_cust
from customers;
```

Màn hình xuất ra:

	num_cust
1	3

Bởi vì trong bảng Products chỉ có ba khách hàng có địa chỉ email.

3. HÀM MAX():

Hàm này sẽ cho ra giá trị lớn nhất của trường, các đối tượng không mang giá trị (null values) sẽ được lờ đi. Ngoài ra hàm max() còn áp dụng cho trường không chứa số (numeric) lúc này hàm max() sẽ trả về dòng cuối cùng của bảng, cú pháp:

```
select max(pro_price) as max_price
from products;
```

Màn hình xuất ra:

	max_price
1	11.99

4. HÀM MIN():

Hàm này sẽ cho ra giá trị nhỏ nhất của trường, các đối tượng không mang giá trị (null values) sẽ được lờ đi. Ngoài ra hàm min() còn áp dụng cho trường không chứa số (numeric) lúc này hàm min() sẽ trả về dòng đầu tiên của bảng, cú pháp:

```
select min(pro_price) as min_price
from products;
```

Màn hình xuất ra:

	min_price
1	3.49

5. HÀM SUM():

Hàm sum() đưa ra giá trị tổng của một trường được chọn với các đối tượng không mang giá trị sẽ được lờ qua, cú pháp:

```
SELECT SUM(QUANTITY) AS ITEMS_ORDERS
FROM ORDERITEMS
WHERE ORDER_NUM=20005;
```

Màn hình xuất ra:

	ITEMS_ORDERS
1	200

Ngoài ra hàm sum() còn tính được các đối tượng của biểu thức, cú pháp:

```
SELECT SUM(QUANTITY*ITEM_PRICE) AS TOTAL_PRICE
FROM ORDERITEMS
WHERE ORDER_NUM=20005;
```

Màn hình xuất ra:

	TOTAL_PRICE
1	1648.00

Lúc này hàm sum() sẽ tính tổng của sản lượng các mặt hàng (quantity) nhân với giá từng mặt hàng có mã số mặt hàng là 20005.

6. HÀM TÍNH TRUNG BÌNH:

Nếu bạn có nhu cầu tính đơn giá trung bình của các sản phẩm được chỉ định trong kì này thì cú pháp như sau:

```
SELECT AVG(DISTINCT PRO_PRICE) AS AVG_PRICE
FROM PRODUCTS
WHERE VEND_ID='DLL01';
```

Màn hình xuất ra:

	AVG_PRICE
1	4.240000

Cách nối các hàm lại với nhau, cú pháp:

```
select count(*) as num_items,
min(pro_price) as price_min,
max(pro_price) as price_max,
avg(pro_price) as price_avg
from products;
```

Màn hình xuất ra:

	num_items	price-min	price-max	price_avg
1	9	3.49	11.99	6.823333

BÀI 10

TẠO NHÓM DỮ LIỆU

Trong bài học này bạn sẽ học cách tạo nhóm dữ liệu có tính tương đồng nhau trong một bảng làm một nhóm (record) với hai mệnh đề mới đó là: Group by và Having.

1. HIỂU VỀ TẠO NHÓM:

Ở bài trước ta đã có thể hình dung được cách tạo nhóm các đối tượng tương đồng nhau để từ đó đưa ra một đại biểu đại diện cho các đối tượng tương đồng đó.

Ví dụ:

```
SELECT COUNT(*) AS NUM_PRODS
FROM PRODUCTS
WHERE VEND_ID='DLL01';
```

Màn hình xuất ra:

	NUM_PRODS
1	4

Trong ví dụ trên bạn đã đếm (count) số bản ghi (records) có mã là DLL01 để lưu vào trường num_prods hay nói cách khác là ta đã nhóm các 'DLL01' lại với nhau.

2. TẠO NHÓM:

Bây giờ bạn sẽ dùng mệnh đề **Group by** để tạo nhóm cho các đối tượng.

Ví dụ:

```
SELECT VEND_ID,COUNT(*) AS NUM_PRODS
FROM PRODUCTS
GROUP BY VEND_ID;
```

Màn hình xuất ra:

	VEND_ID	NUM_PRODS
1	BRS01	2
2	DLL01	4
3	FNG01	2

Trong ví dụ trên bạn sẽ thấy mệnh đề **Group by** đã nhóm các mã số đại lí (vend_id) giống nhau lại thành các **records** và cho vào trường **num_prods**.

3. BỘ LỌC NHÓM:

Sau đây bạn sẽ học được cách dùng mệnh đề HAVING để lọc các nhóm đã tạo ra.

Ví dụ:

```
SELECT CUST_ID,COUNT(*) AS ORDERS
FROM ORDERS
GROUP BY CUST_ID
HAVING COUNT(*)>=2;
```

Màn hình xuất ra:

	VEND_ID	ORDERS
1	1000000001	2

Trong ví dụ trên bạn đã lọc được các nhóm mã số khách hàng (cust_id) là orders lớn hơn hay bằng 2.

Phân biệt giữa hai mệnh đề HAVING và WHERE: mệnh đề **Where** lọc dữ liệu trước khi được nhóm còn mệnh đề **Having** thì

lọc dữ liệu sau khi được nhóm.

Ví dụ:

```
SELECT VEND_ID,COUNT(*) AS NUM_PRODS
FROM PRODUCTS
WHERE PROD_PRICE>=4
GROUP BY VEND_ID
HAVING COUNT(*)>=2;
```

Màn hình xuất ra:

	VEND_ID	NUM_PRODS
1	BRS01	2
2	FNG01	2

Trong ví dụ trên ta thấy mệnh đề **where** đã lọc giá sản phẩm (prod_price) lớn hơn hoặc bằng 4 trước, sau đó ta mới tạo nhóm các **vend_id** và mệnh đề **having** sẽ làm công việc cuối cùng là lọc các nhóm đó. Tuy nhiên nếu như trong ví dụ trên ở các dòng lệnh bạn bỏ đi mệnh đề lọc **where** thì kết quả sẽ khác:

```
SELECT VEND_ID,COUNT(*) AS NUM_PRODS
FROM PRODUCTS
GROUP BY VEND_ID
HAVING COUNT(*)>=2;
```

Màn hình xuất ra:

	VEND_ID	NUM_PRODS
1	BRS01	2
2	DLL01	4
3	FNG01	2

4. PHÂN BIỆT HAI MỆNH ĐỀ GROUP BY VÀ ORDER BY:

ORDER BY	GROUP BY
Sắp xếp đầu ra.	Nhóm dòng. Đầu ra có thể không cùng nhóm.
Sắp xếp tất cả các cột.	Chỉ nhóm các cột được chọn.
Không bắt buộc.	Phải có khi sử dụng hàm tổng.

Ví dụ:

```
SELECT ORDER_NUM,COUNT(*) AS ITEMS
FROM ORDERITEMS
GROUP BY ORDER_NUM
HAVING COUNT(*)>=3;
```

Màn hình xuất ra:

	ORDER_NUM	ITEMS
1	20006	3
2	20007	5
3	20008	5
4	20009	3

Trong ví dụ trên ta đã tạo nhóm và lọc nhóm, bây giờ ta sẽ sắp xếp các đối tượng đầu ra với mệnh đề **order by**:

```
SELECT ORDER_NUM,COUNT(*) AS ITEMS
FROM ORDERITEMS
GROUP BY ORDER_NUM
HAVING COUNT(*)>=3
ORDER BY ITEMS,ORDER_NUM;
```

Màn hình xuất ra:

	ORDER_NUM	ITEMS
1	20006	3
2	20009	3
3	20007	5
4	20008	5

Bảng ôn lại các mệnh đề:

MỆNH ĐỀ	DIỄN GIẢI	YÊU CẦU
Select	Cột hay phần mở rộng được trả lại.	Có.
From	Bảng để truy vấn dữ liệu.	Chỉ khi nào truy vấn dữ liệu từ bảng.
Where	Lọc dòng.	Không.
Group by	Tạo nhóm.	Chỉ đi kèm hàm tổng.
Having	Lọc nhóm.	Không.
Oder by	Sắp xếp đầu ra.	Không.

BÀI 11

LÀM VIỆC VỚI TỪ TRUY VẤN

Trong bài học này bạn sẽ học về các câu lệnh với từ truy vấn, nó sẽ giúp bạn tốt hơn trong việc tìm kiếm dữ liệu.

1. LỌC DỮ LIỆU VỚI TỪ TRUY VẤN:

Với mệnh đề 'Where' bạn cũng có thể lọc được các đối tượng nằm trong bảng.

Ví dụ:

```
SELECT ORDER_NUM
FROM ORDERITEMS
WHERE PROD_ID='RGAN01';
```

Màn hình xuất ra:

	ORDER_NUM
1	20007
2	20008

Trong ví dụ trên bạn vừa truy xuất các mã số sản phẩm (prod_id) là RGAN01 từ bảng orderitems. Bây giờ bạn cần truy xuất mã số khách hàng (cust_id) của các order_num 20007&20008, câu lệnh:

```
SELECT CUST_ID
FROM ORDERS
WHERE ORDER_NUM IN (20007,20008);
```

Màn hình xuất ra:

	CUST_ID
1	1000000004
2	1000000005

Khi đó bạn thấy từ truy vấn 'IN' sẽ lọc các cust_id thuộc (20007, 20008). Nếu bạn thực hiện nối (combine) các câu lệnh trên lại cũng cho kết quả tương tự, câu lệnh:

```
SELECT CUST_ID
FROM ORDERS
WHERE ORDER_NUM IN
(
SELECT ORDER_NUM
FROM ORDERITEMS
WHERE PROD_ID='RGAN01'
);
```

Trong dòng lệnh trên bạn lưu ý đoạn **subquery** sẽ được thực hiện trước:

```
(SELECT ORDER_NUM
FROM ORDERITEMS
WHERE PROD_ID='RGAN01')
```

Ta có thể lọc ở các cấp độ sâu hơn như sau:

Bạn đã có mã số (IDs) của tất cả các khách hàng thuộc (1000000004, 1000000005). Bước kế tiếp thông tin khách hàng cho từng IDs này, câu lệnh:

```
SELECT CUST_NAME,CUST_CONTACT
FROM CUSTOMERS
WHERE CUST_ID IN ('1000000004','1000000005');
```

Hay:

```
SELECT CUST_NAME,CUST_CONTACT
FROM CUSTOMERS
WHERE CUST_ID IN
(
SELECT CUST_ID
FROM ORDERS
WHERE ORDER_NUM IN
(
SELECT ORDER_NUM
FROM ORDERITEMS
WHERE PROD_ID='RGAN01'
)
);
```

Màn hình xuất ra:

	CUST_NAME	CUST_CONTACT
1	Fun4All	Denise L. Stephens
2	The Toy Store	Kim Howard

2. DÙNG SUBQUERY NHƯ MỘT TRƯỜNG TÍNH TOÁN:

Một cách khác người ta sử dụng subquery như một caculated fields.

Ví dụ:

Với mục đích bạn muốn hiển thị tổng số hóa đơn của mỗi khách hàng có trong bảng khách hàng (customers table). Còn hóa đơn thì được đặt trong bảng hóa đơn (orders table) với các customer ID. Để làm được việc trên thì bạn phải làm các bước tuần tự sau:

- Truy xuất danh sách khách hàng từ bảng khách hàng.
- Ứng với mỗi khách hàng truy xuất, tổng số hóa đơn của họ trong orders table.

```
SELECT CUST_NAME,CUST_STATE,
(SELECT COUNT(*)
FROM ORDERS
```

```
WHERE ORDERS.CUST_ID=CUSTOMERS.CUST_ID
) AS ORDERS
FROM CUSTOMERS
ORDER BY CUST_NAME;
```

Màn hình xuất ra:

	CUST_NAME	CUST_STATE	ORDERS
1	Fun4All	IN	1
2	Fun4All	AZ	1
3	Kids Place	OH	0
4	The Toy Store	IL	1
5	Village Toys	MI	2

Chú giải:

ORDERS.CUST_ID: đề cập đến các cust_id có trong bảng orders.

CUSTOMERS.CUST_ID: đề cập đến các cust_id có trong bảng customers.

Cũng ví dụ trên bạn so sánh các cust_id của orders table với nhau thì kết quả là:

```
SELECT CUST_NAME,CUST_STATE,
(SELECT COUNT(*)
FROM ORDERS
WHERE CUST_ID=CUST_ID
) AS ORDERS
FROM CUSTOMERS
ORDER BY CUST_NAME;
```

Màn hình xuất ra:

	CUST_NAME	CUST_STATE	ORDERS
1	Fun4All	IN	5
2	Fun4All	AZ	5
3	Kids Place	OH	5
4	The Toy Store	IL	5
5	Village Toys	MI	5

BÀI 12

LIÊN KẾT BẢNG

Trong bài học này bạn sẽ học cách liên kết bảng, các chức năng của chúng cũng như cách dùng câu lệnh chọn để thực hiện chúng.

Như các bạn đã biết các bảng trong lập trình cơ sở dữ liệu luôn có sự liên kết (relation) và có thể trao đổi dữ liệu lẫn nhau. Bạn có thể lấy một ví dụ cho bài học của chúng ta, bây giờ bạn hãy tạo hai bảng một bảng thông tin về các đại lí (vendors) còn một bảng thông tin về các sản phẩm (products). Trong bảng vendors bao gồm nhiều vendor khác nhau, trên mỗi dòng (row, record) là một vendor với một sự định dạng nhất định. Giá trị định dạng này được gọi là một khóa chính (***Primary key***), nó có thể là một vendor ID hay các giá trị khác.

Trong bảng sản phẩm nó lưu trữ các thông tin về sản phẩm. Khóa chính sẽ liên kết Vendors table với Products table lại với nhau.

1. TẠO MỘT LIÊN KẾT:

Nhìn chung để tạo một liên kết rất đơn giản. Bạn chỉ việc xác định rõ các bảng và cách thức chúng liên kết nhau (thông qua Primary key).

Ví dụ:

Bây giờ bạn hãy tìm các vend_id của cả hai bảng vendors và products sao cho chúng trùng nhau. Đầu tiên bạn hãy xem tổng thể cả hai bảng:

```
SELECT *
FROM PRODUCTS
```

Màn hình xuất ra:

	PROD_ID	VEND_ID	PROD_NAME	PROD_PRICE	PROD_DESC
1	BR01	BRS01	8 inch tedddy bear	5.99	8 inch teddy bear, comes with cap and jacket
2	BR02	BRS01	12 inch teddy bear	8.99	12 inch teddy bear, comes with cap and jacket
3	BNBG01	DLL01	Fish bean bag toy	3.49	Fish bean bag toy, complete with bean bag worms with which to feed it
4	BNBG02	DLL01	Bird bean bag toy	3.49	Bird bean bag toy, eggs are not included
5	BNBG03	DLL01	Rabbit bean bag toy	3.49	Rabbit bean bag toy, comes with bean bag carrots
6	RGAN01	DLL01	Raggedy Ann	4.99	18 inch Raggedy Ann doll
7	RYL01	FNG01	King doll	9.49	12 inch king doll with royal garments and crown
8	RYL02	FNG01	Queen doll	9.49	12 inch king doll with royal garments and crown

```
SELECT *
FROM VENDORS;
```

Màn hình xuất ra:

	VEND_ID	VEND_NAME	VEND_ADDRESS	VEND_CITY	VEND_STATE	VEND_ZIP	VEND_COUNTRY
1	BRS01	Bears Rus	123 Main Street	Bear Town	MI	44444	USA
2	BRS01	Bear Emporium	500 Park Street	Any town	OH	44333	USA
3	DLL01	Doll House Inc	555 High Street	Dollsville	CA	99999	USA
4	FRB01	Furball Inc	1000 5th Avenue	New York	NY	11111	USA
5	FNG01	Fun and Games	42 Galaxy Road	London		N16 6ps	England
6	JTS01	Jouets et ours	1 Rue Amusement	Paris		45678	France

Câu lệnh so sánh các vend_id của hai bảng vendors và products như sau:

```
SELECT VEND_NAME,PROD_NAME,PROD_PRICE
FROM VENDORS,PRODUCTS
WHERE VENDORS.VEND_ID=PRODUCTS.VEND_ID
ORDER BY VEND_NAME DESC;
```

Màn hình xuất ra:

	VEND_NAME	PROD_NAME	PROD_PRICE
1	Fun and Games	King doll	9.49
2	Fun and Games	Queen doll	9.49
3	Doll House Inc	Fish bean bag toy	3.49
4	Doll House Inc	Bird bean bag toy	3.49
5	Doll House Inc	Rabbit bean bag toy	3.49
6	Doll House Inc	Raggedy Ann	4.99
7	Bears Rus	8 inch teddy bear	5.99
8	Bears Rus	12 inch teddy bear	8.99

Thông qua ví dụ trên các bạn đã thấy hai bảng vendors và products đã truy xuất dữ liệu lẫn nhau để xuất ra bảng liên kết (joining table).

Sự quan trọng của mệnh đề '**Where**': có thể nói mệnh đề where đóng một vai trò rất quan trọng trong việc tạo liên kết bảng bởi vì với vai trò một bộ lọc dữ liệu động **'where'** đã giúp cho ta so sánh các key words trong các bảng khác nhau. Một ví dụ sau sẽ mô tả nếu không có mệnh đề **where**, cú pháp:

```
SELECT VEND_NAME,PROD_NAME,PROD_PRICE
FROM VENDORS,PRODUCTS;
```

Màn hình xuất ra:

	VEND_NAME	PROD_NAME	PROD_PRICE
1	Bears Rus	8 inch teddy bear	5.99
2	Bears Rus	12 inch teddy bear	8.99
3	Bears Rus	Fish bean bag toy	3.49
4	Bears Rus	Bird bean bag toy	3.49
5	Bears Rus	Rabbit bean bag toy	3.49
6	Bears Rus	Raggedy Ann	4.99
7	Bears Rus	King doll	9.49
8	Bears Rus	Queen doll	9.49
9	Bear Emporium	8 inch teddy bear	5.99
10	Bear Emporium	12 inch teddy bear	8.99
11	Bear Emporium	Fish bean bag toy	3.49
12	Bear Emporium	Bird bean bag toy	3.49
13	Bear Emporium	Rabbit bean bag toy	3.49
14	Bear Emporium	Raggedy Ann	4.99
15	Bear Emporium	King doll	9.49
16	Bear Emporium	Queen doll	9.49
17	Doll House Inc.	8 inch teddy bear	5.99
18	Doll House Inc.	12 inch teddy bear	8.99
19	Doll House Inc.	Fish bean bag toy	3.49
20	Doll House Inc.	Bird bean bag toy	3.49
21	Doll House Inc.	Rabbit bean bag toy	3.49
22	Doll House Inc.	Raggedy Ann	4.99
23	Doll House Inc.	King doll	9.49
24	Doll House Inc.	Queen doll	9.49
25	Furball Inc.	8 inch teddy bear	5.99
26	Furball Inc.	12 inch teddy bear	8.99
27	Furball Inc.	Fish bean bag toy	3.49
28	Furball Inc.	Bird bean bag toy	3.49

	VEND_NAME	PROD_NAME	PROD_PRICE
29	Furball Inc.	Rabbit bean bag toy	3.49
30	Furball Inc.	Raggedy Ann	4.99
31	Furball Inc.	King doll	9.49
32	Furball Inc.	Queen doll	9.49
33	Fun and Games	8 inch teddy bear	5.99
34	Fun and Games	12 inch teddy bear	8.99
35	Fun and Games	Fish bean bag toy	3.49
36	Fun and Games	Bird bean bag toy	3.49
37	Fun and Games	Rabbit bean bag toy	3.49
38	Fun and Games	Raggedy Ann	4.99
39	Fun and Games	King doll	9.49
40	Fun and Games	Queen doll	9.49
41	Jouets et ours	8 inch teddy bear	5.99
42	Jouets et ours	12 inch teddy bear	8.99
43	Jouets et ours	Fish bean bag toy	3.49
44	Jouets et ours	Bird bean bag toy	3.49
45	Jouets et ours	Rabbit bean bag toy	3.49
46	Jouets et ours	Raggedy Ann	4.99
47	Jouets et ours	King doll	9.49
48	Jouets et ours	Queen doll	9.49

2. TẠO LIÊN KẾT NỘI BỘ (INNER JOINS):

Nó cũng giống như việc tạo liên kết giữa 2 bảng.

Ví dụ:

```
SELECT VEND_NAME,PROD_NAME,PROD_PRICE
FROM VENDORS
INNER JOIN PRODUCTS
ON VENDORS.VEND_ID=PRODUCTS.VEND_ID;
```

Màn hình xuất ra:

	VEND_NAME	PROD_NAME	PROD_PRICE
1	Bears Rus	8 inch teddy bear	5.99
2	Bears Rus	12 inch teddy bear	8.99
3	Doll House Inc.	Fish bean bag toy	3.49
4	Doll House Inc.	Bird bean bag toy	3.49
5	Doll House Inc.	Rabbit bean bag toy	3.49
6	Doll House Inc.	Raggedy Ann	4.99
7	Fun and Games	King doll	9.49
8	Fun and Games	Queen doll	9.49

3. LIÊN KẾT NHIỀU BẢNG:

SQL cho phép bạn tạo liên kết các bảng lại với nhau không giới hạn, song vẫn dựa trên nguyên tắc (rules) chung.

Ví dụ:

```
SELECT PROD_NAME,VEND_NAME,PROD_PRICE,QUANTITY
FROM ORDERITEMS,PRODUCTS,VENDORS
WHERE PRODUCTS.VEND_ID=VENDORS.VEND_ID
AND ORDERITEMS.PROD_ID=PRODUCTS.PROD_ID
AND ORDER_NUM=20007;
```

Màn hình xuất ra:

	PROD_NAME	VEND_NAME	PROD_PRICE	QUANTITY
1	Raggedy Ann	Doll House Inc.	4.99	50
2	Fish bean bag toy	Doll House Inc.	3.49	100
3	Bird bean bag toy	Doll House Inc.	3.49	100
4	Rabbit bean bag toy	Doll House Inc.	3.49	100

Trong ví dụ trên ta thấy ba bảng orderitems, products và vendors đã so sánh các key words với nhau. Nhìn chung khi các bạn dùng liên kết nhiều bảng lại với nhau thì câu lệnh sẽ bớt phức tạp hơn, mang tính chính xác cao:

Ví dụ:

```
SELECT CUST_NAME,CUST_CONTACT
FROM CUSTOMERS,ORDERS,ORDERITEMS
WHERE CUSTOMERS.CUST_ID=ORDERS.CUST_ID
AND ORDERITEMS.ORDER_NUM=ORDERS.ORDER_NUM
AND PROD_ID='RGAN01';
```

Hoặc:

```
SELECT CUST_NAME,CUST_CONTACT
FROM CUSTOMERS
WHERE CUST_ID IN
(
SELECT CUST_ID
FROM ORDERS
WHERE ORDER_NUM IN
(
SELECT ORDER_NUM
FROM ORDERITEMS
WHERE PROD_ID='RGAN01'
)
);
```

Màn hình xuất ra:

	CUST_NAME	CUST_CONTACT
1	Fun4All	Denise L. Stephens
2	The Toy Store	Kim Howard

BÀI 13

TẠO LIÊN KẾT BẢNG THÔNG MINH

Trong bài học này bạn sẽ được hiểu thêm về một số kiểu liên kết bảng. Ngoài ra bạn cũng sẽ biết cách sử dụng bảng liên kết với các hàm tổng.

1. SỬ DỤNG CÁC ĐỊNH DANH CHO BẢNG:

Như ở các bài học trước các bạn đã học cách sử dụng các định danh (Aliases) để truy xuất dữ liệu từ bảng, nhắc lại cú pháp::

```
SELECT RTRIM(VEND_CITY)+','+RTRIM(VEND_STATE)+'
'+RTRIM(VEND_ZIP) AS ADDRESS2
FROM VENDORS
ORDER BY VEND_NAME;
```

Màn hình xuất ra:

	ADDRESS2
1	Any town, OH 44333
2	Bear Town, MI 44444
3	Dollsville, CA 99999
4	London, N16 6PS
5	New York, NY 11111
6	Paris, 45678

Ngoài ra để sử dụng được Aliases cho các tên cột và các trường tính toán, SQL phải thực hiện định danh tên các cột. Có hai lí do để làm việc này:

- Làm các câu lệnh được rút gọn.
- Làm việc trên nhiều bảng chỉ với một lệnh SELECT.

Bây giờ bạn hãy tham khảo ví dụ dưới đây để hiểu rõ hơn:

```
SELECT CUST_NAME,CUST_CONTACT
FROM CUSTOMERS AS C,ORDERS AS O,ORDERITEMS AS OI
WHERE C.CUST_ID=O.CUST_ID
AND    OI.ORDER_NUM=O.ORDER_NUM
AND    PROD_ID='RGAN01';
```

Màn hình xuất ra:

	CUST_NAME	CUST_CONTACT
1	Fun4All	Denise L. Stephens
2	The Toy Store	Kim Howard

Trong ví dụ trên bạn dễ thấy 3 bảng trong mệnh đề FROM tất cả đều có Aliases lần lượt là: C, O và OI. Nó là các Aliases là các kí tự đầu tiên đại diện cho toàn bộ các tables.

Lưu ý:

Trong SQL cú pháp câu lệnh phải có từ khóa 'AS' còn trong cú pháp của câu lệnh Oracle hay các trình cơ sở dữ liệu có hỗ trợ (support) 'DBMS' thì không cần, cú pháp:

```
SELECT CUST_NAME,CUST_CONTACT
FROM CUSTOMERS  C,ORDERS  O,ORDERITEMS  OI
WHERE C.CUST_ID=O.CUST_ID
AND   OI.ORDER_NUM=O.ORDER_NUM
AND   PROD_ID='RGAN01';
```

Một lưu ý nữa là các table Aliases chỉ là các đơn vị ảo, đây là điểm khác biệt với các column aliases.

2. CÁC KIỂU LIÊN KẾT:

a) Tạo các tự liên kết (Self Joins):

Như đã được đề cập ưu điểm của sử dụng table aliases là có thể đưa các bảng giống nhau nhiều về các yếu tố chỉ với một câu lệnh SELECT.

Với mục đích bạn muốn email tới những người làm cùng một công ty chẳng hạn là **Jim Jones**. Điều này yêu cầu bạn trước tiên phải tìm cho được công ty của họ đang làm việc và tiếp theo bạn phải biết có những **customers** nào đang làm việc cho công ty đó. Sau đây là một trong số các cách giải quyết vấn đề trên, cú pháp câu lệnh:

```
SELECT CUST_ID,CUST_NAME,CUST_CONTACT
FROM CUSTOMERS
WHERE CUST_NAME=
        (
        SELECT CUST_NAME
        FROM CUSTOMERS
        WHERE CUST_CONTACT='JIM JONES'
        );
```

Màn hình xuất ra:

	CUST_ID	CUST_NAME	CUST_CONTACT
1	1000000003	Fun4All	Jim Jones
2	1000000004	Fun4All	Denise L. Stephens

Bây giờ bạn hãy xem cấu trúc của câu lệnh **aliases join** để thực hiện việc trên với cú pháp **join**:

```
SELECT C1.CUST_ID,C1.CUST_NAME,C1.CUST_CONTACT
FROM CUSTOMERS AS C1,CUSTOMERS AS C2
WHERE C1.CUST_NAME=C2.CUST_NAME
AND   C2.CUST_CONTACT='JIM JONES';
```

b) Liên kết thông thường (natural joins):

Thật ra natural joins là một liên kết mà bạn chỉ thực hiện độc nhất trên các cột (columns). Đây là kiểu thực hiện có sử dụng wildcard (kí tự đại diện) 'SELECT *' cho một bảng và các cột sẽ được truy xuất bởi các bảng khác.

Ví dụ:

```
SELECT C.*,O.ORDER_NUM,O.ORDER_DATE,OI.PROD_ID,
OI.QUANTITY,OI.ITEM_PRICE
FROM CUSTOMERS AS C,ORDERS AS O,ORDERITEMS AS OI
WHERE C.CUST_ID=O.CUST_ID
AND   OI.ORDER_NUM=O.ORDER_NUM
AND   PROD_ID='RGAN01';
```

Nếu bạn dùng trình cơ sở dữ liệu Oracle thì cú pháp câu lệnh như sau:

```
SELECT
C.*,O.ORDER_NUM,O.ORDER_DATE,OI.PROD_ID,
OI.QUANTITY,OI.ITEM_PRICE
FROM CUSTOMERS C,ORDERS O,ORDERITEMS OI
WHERE C.CUST_ID=O.CUST_ID
AND   OI.ORDER_NUM=O.ORDER_NUM
AND   PROD_ID='RGAN01';
```

Màn hình xuất ra:

	cust_id	cust_name	cust_address	cust-city
1	1000000004	Fun4All	829 Riverside Drive	Phoenix
2	1000000005	The Toy Store	4545 53rd Street	Chicago

cust_state	cust_zip	cust_country	cust_contact	cust_email
AZ	88888	USA	Denise L. Stephens	dstephens@fun4all.com
IL	54545	USA	Kim Howard	

ODER_NUM	ODER_DATE	PROD_ID	QUANTITY	ITEM_PRICE
2007	2004_01_30 00:00:00.000	RGAN01	50	4.49
2008	2004_02_03 00:00:00.000	RGAN01	5	4.99

c) Liên kết ngoài (Outer joins):

Hầu hết các dòng trong một bảng liên hệ với những dòng khác trong các bảng khác. Tuy nhiên, nếu bạn muốn tìm các dòng không có mối liên hệ, chẳng hạn như:

- Bạn tính xem mỗi khách hàng có bao nhiêu hóa đơn, kể cả những khách hàng chưa có hóa đơn.
- Danh sách tất cả các sản phẩm với số lượng có trước được đặt hàng, kể cả những sản phẩm chưa được đặt hàng.
- Tính tổng lại sau đó đưa vào một tài khoản chưa được đặt hàng.

Đầu tiên bạn nên truy xuất tất cả các khách hàng đã được đặt hàng, câu lệnh:

```
SELECT CUSTOMERS.CUST_ID,ORDERS.ORDER_NUM
FROM CUSTOMERS,ORDERS
WHERE CUSTOMERS.CUST_ID=ORDERS.CUST_ID;
```

Hoặc:

```
SELECT CUSTOMERS.CUST_ID,ORDERS.ORDER_NUM
FROM CUSTOMERS,ORDERS
WHERE CUSTOMERS.CUST_ID=*ORDERS.CUST_ID;
```

Hoặc:

```
SELECT CUSTOMERS.CUST_ID,ORDERS.ORDER_NUM
FROM CUSTOMERS RIGHT OUTER JOIN ORDERS
ON CUSTOMERS.CUST_ID=ORDERS.CUST_ID;
```

Màn hình xuất ra:

	CUST_ID	ORDER_NUM
1	1000000003	20006
2	1000000001	20005
3	1000000004	20007
4	1000000005	20008
5	1000000001	20009

Tiếp theo bạn có thể truy xuất tất cả các khách hàng không đặt hàng bằng cách sau:

```
SELECT CUSTOMERS.CUST_ID,ORDERS.ORDER_NUM
FROM CUSTOMERS,ORDERS
WHERE CUSTOMERS.CUST_ID*=ORDERS.CUST_ID;
```

Hay:

```
SELECT CUSTOMERS.CUST_ID,ORDERS.ORDER_NUM
FROM CUSTOMERS LEFT OUTER JOIN ORDERS
ON CUSTOMERS.CUST_ID=ORDERS.CUST_ID;
```

Màn hình xuất ra:

	CUST_ID	ORDER_NUM
1	1000000001	20005
2	1000000001	20009
3	1000000002	NULL
4	1000000003	20006
5	1000000004	20007
6	1000000005	20008

Ngoài ra, nếu bạn sử dụng trình cơ sở dữ liệu **Oracle** thì cú pháp như sau:

```
SELECT CUSTOMERS.CUST_ID,ORDERS.ORDER_NUM
FROM CUSTOMERS,ORDERS
WHERE CUSTOMERS.CUST_ID(+)=ORDERS.CUST_ID;
```

Bạn còn có thể dùng một cú pháp khác để xem các khách hàng đã đặt hóa đơn và kể cả chưa đặt hóa đơn, câu lệnh:

```
SELECT CUSTOMERS.CUST_ID,ORDERS.ORDER_NUM
FROM CUSTOMERS FULL OUTER JOIN ORDERS
ON CUSTOMERS.CUST_ID=ORDERS.CUST_ID;
```

Màn hình xuất ra:

	CUST_ID	ORDER_NUM
1	1000000003	20006
2	1000000001	20005
3	1000000004	20007
4	1000000005	20008
5	1000000001	20009
6	1000000002	NULL

3. SỬ DỤNG LIÊN KẾT VỚI CÁC HÀM TRUNG BÌNH:

Giả sử bây giờ bạn cần truy xuất danh sách của tất cả khách hàng và số hóa đơn của họ. Lúc này bạn sẽ phải sử dụng hàm Count() để thực hiện điều này:

```
SELECT CUSTOMERS.CUST_ID,
COUNT(ORDERS.ORDER_NUM) AS NUM_PRO
FROM CUSTOMERS INNER JOIN ORDERS
ON CUSTOMERS.CUST_ID=ORDERS.CUST_ID
GROUP BY CUSTOMERS.CUST_ID;
```

Màn hình xuất ra:

	CUST_ID	NUM_PRO
1	1000000001	2
2	1000000003	1
3	1000000004	1
4	1000000005	1

Trong ví dụ trên bạn đã dùng cụm từ 'INNER JOIN' để liên kết hai bảng customers và bảng orders. Còn nếu bạn dùng OUTER

JOIN thì nó sẽ lọc ra kể cả các khách hàng không có hóa đơn:

```
SELECT CUSTOMERS.CUST_ID,
COUNT(ORDERS.ORDER_NUM) AS NUM_PRO
FROM CUSTOMERS LEFT OUTER JOIN ORDERS
ON CUSTOMERS.CUST_ID=ORDERS.CUST_ID
GROUP BY CUSTOMERS.CUST_ID;
```

Màn hình xuất ra:

	CUST_ID	NUM_PRO
1	1000000001	2
2	1000000002	0
3	1000000003	1
4	1000000004	1
5	1000000005	1

BÀI 14

NỐI CÁC CÂU TRUY VẤN

Trong bài học này bạn sẽ học được cách để sử dụng toán tử UNION để nối nhiều câu lệnh chọn SELECT lại với nhau để xuất ra một bảng.

Tạo nối câu truy vấn.

Sử dụng từ nối UNION.

Ví dụ bạn cần biết tất cả các khách hàng ở Illinois, Indiana và Michigan. Ngoài ra bạn cũng muốn biết cust_name là 'Fun4all' nữa, theo mệnh đề 'Where' thì bạn phải làm hai động tác sau:

```
SELECT CUST_NAME,CUST_CONTACT,CUST_EMAIL
FROM CUSTOMERS
WHERE CUST_STATE IN ('IL','IN','MI');
```

Màn hình xuất ra:

	CUST_NAME	CUST_CONTACT	CUST_EMAIL
1	Village Toys	John Smith	sales@villagetoys.com...
2	Fun4All	Jim Jones	jjones@fun4all.com
3	The Toy Store	Kim Howard	

Và:

```
SELECT CUST_NAME,CUST_CONTACT,CUST_EMAIL
FROM CUSTOMERS
WHERE CUST_NAME='Fun4all';
```

Màn hình xuất ra:

	CUST_NAME	CUST_CONTACT	CUST_EMAIL
1	Fun4All	Jim Jones	jones@fun4all.com
2	Fun4All	Denise L. Stephens	dstephens@fun4all.com...

Còn nếu bạn dùng từ nối để thực hiện yêu cầu trên thì nó sẽ xuất ra trong một bảng:

```
SELECT CUST_NAME,CUST_CONTACT,CUST_EMAIL
FROM CUSTOMERS
WHERE CUST_STATE IN ('IL','IN','MI')
UNION
SELECT CUST_NAME,CUST_CONTACT,CUST_EMAIL
FROM CUSTOMERS
WHERE CUST_NAME='Fun4all';
```

Màn hình xuất ra:

	CUST_NAME	CUST_CONTACT	CUST_EMAIL
1	Fun4All	Denise L. Stephens	dstephens@fun4all.com
2	Fun4All	Jim Jones	jjones@fun4all.com
3	The Toy Store	Kim Howard	
4	Viilage Toys	John Smith	sales@villagetoys.com

Ngoài ra nếu bạn sử dụng nhiều lần mệnh đề 'Where' cũng cho ra kết quả tương tự:

```
SELECT CUST_NAME,CUST_CONTACT,CUST_EMAIL
FROM CUSTOMERS
WHERE CUST_STATE IN ('IL','IN','MI')
OR
CUST_NAME='FUN4ALL';
```

Tuy nhiên từ nối **'UNION'** khi được sử dụng sẽ không giới hạn table và đỡ rắc rối hơn khi có nhiều bảng cần được truy xuất so với mệnh đề **'Where'**.

Nếu bạn sử dụng từ nối **'UNION ALL'** nó sẽ thực hiện nối tất cả các columns và không sử dụng luật loại trừ, tránh trùng lặp:

```
SELECT CUST_NAME,CUST_CONTACT,CUST_EMAIL
FROM CUSTOMERS
WHERE CUST_STATE IN ('IL','IN','MI')
UNION ALL
```

```
SELECT CUST_NAME,CUST_CONTACT,CUST_EMAIL
FROM CUSTOMERS
WHERE CUST_NAME='Fun4all';
```

Màn hình xuất ra:

	CUST_NAME	CUST_CONTACT	CUST_EMAIL
1	Viilage Toys	John Smith	sales@villagetoys.com
2	Fun4All	Jim Jones	jjones@fun4all.com
3	The Toy Store	Kim Howard	
4	Fun4All	Jim Jones	jjones@fun4all.com
5	Fun4All	Denise L. Stephens	dstephens@fun4all.com

Sắp xếp kết quả nối câu truy vấn:

Ta chỉ có thể dùng một mệnh đề **'Order by'** để sắp xếp kết quả cuối cùng của câu truy vấn dùng từ nối 'UNION'.

Ví dụ:

```
SELECT CUST_NAME,CUST_CONTACT,CUST_EMAIL
FROM CUSTOMERS
WHERE CUST_STATE IN ('IL','IN','MI')
UNION ALL
SELECT CUST_NAME,CUST_CONTACT,CUST_EMAIL
FROM CUSTOMERS
WHERE CUST_NAME='Fun4all'
ORDER BY CUST_NAME,CUST_CONTACT;
```

Màn hình xuất ra:

	CUST_NAME	CUST_CONTACT	CUST_EMAIL
1	Fun4All	Denise L. Stephens	dstephens@fun4all.com
2	Fun4All	Jim Jones	jjones@fun4all.com
3	Fun4All	Jim Jones	jjones@fun4all.com
4	The Toy Store	Kim Howard	
5	Viilage Toys	John Smith	sales@villagetoys.com

BÀI 15

CHÈN DỮ LIỆU

Trong bài học này bạn sẽ học cách chèn dữ liệu vào bảng sử dụng câu lệnh INSERT trong cấu trúc câu lệnh chọn SQL.

Như bạn được biết câu lệnh INSERT dùng để chèn dữ liệu vào các dòng của bảng cơ sở dữ liệu, nó được sử dụng bằng nhiều cách:

- Chèn vào một dòng đã có sẵn (a complete row).
- Chèn vào một phần của dòng (a partial row).
- Chèn vào dòng các kết quả của một câu lệnh truy vấn nào đó (results of a query).

1. CHÈN VÀO MỘT DÒNG ĐÃ CÓ SẴN:

Cách đơn giản nhất để chèn dữ liệu vào một bảng là sử dụng cú pháp lệnh INSERT. Bạn nhất thiết phải biết được tên của bảng được chèn và những giá trị sẽ được chèn vào bảng đó.

Ví dụ:

```
INSERT INTO CUSTOMERS
VALUES
(
'1000000006',
'Toy Land',
'123 Any Street',
'New York',
'NY',
'11111',
```

'USA',
'',
''
);

Màn hình xuất ra:

	cust_id	cust_name	cust_address	cust-city
1	1000000001	Viilage Toys	200 Maple Lane	Detroit
2	1000000002	Kids Place	333 South Lake Drive	Columbus
3	1000000003	Fun4All	1 Sunny Place	Muncie
4	1000000004	Fun4All	829 Riverside Drive	Phoenix
5	1000000005	The Toy Store	4545 53rd Street	Chicago
6	1000000006	Toy Land	123 Any Street	New York

	cust_state	cust_zip	cust_country	cust_contact	cust_email
1	MI	44444	USA	John Smith	sales@villagetoys.com
2	OH	43333	USA	Michelle Green	
3	IN	42222	USA	Jim Jones	jjones@fun4all.com
4	AZ	88888	USA	Denise L. Stephens	dstephens@fun4all.com
5	IL	54545	USA	Kim Howard	
6	NY	11111	USA		

Lúc này trong bảng 'Customers' bạn sẽ thấy xuất hiện thêm dòng thứ 6:

6	1000000006	Toy Land	123 Any Street	New York
6	NY	11111	USA	

Trong ví dụ trên ta đã chèn một khách hàng mới vào trong bảng customers. Dữ liệu sẽ được lưu vào trong mỗi cột của bảng đó là các giá trị nằm sau mệnh đề 'VALUES' và lúc này các dữ liệu đó sẽ được cập nhật vào các cột tương ứng. Ngoài ra các cột

không có dữ liệu tức cột rỗng sẽ được nhận giá trị 'NULL'. Mặc dù vậy cách nhập dữ liệu này không được an toàn cho lắm vì ta có thể nhầm các giá trị của các cột khác nhau, sau đây sẽ cung cấp bạn một cách cập nhật dữ liệu khác cho kết quả như trên:

```
        INSERT INTO CUSTOMERS
(
        cust_id,
        cust_name,
        cust_address,
        cust_city,
        cust_state,
        cust_zip,
        cust_country,
        cust_contact,
        cust_email
)
VALUES
(
        '1000000006',
        'Toy Land',
        '123 Any Street',
        'New York',
        'NY',
        '11111',
        'USA',
        '',
        ''
);
```

2. CHÈN VÀO TỪNG PHẦN CỦA DÒNG:

Trong một dòng có rất nhiều cột nếu như bạn không muốn cập nhật dữ liệu cho tất cả các cột mà chỉ update cho một số cột bạn có thể thực hiện như sau:

```
INSERT INTO CUSTOMERS
(
        cust_id,
        cust_name,
        cust_address,
        cust_city,
        cust_state,
        cust_zip,
        cust_country
)
VALUES
(
        '1000000006',
        'Toy Land',
        '123 Any Street',
        'New York',
        'NY',
        '11111',
        'USA'
);
```

Trong ví dụ trên bạn đã không cập nhật dữ liệu vào hai cột 'cust_contact' và 'cust_email'. Lúc này giá trị mặc định của nó là 'NULL'.

3. CHÈN KẾT QUẢ DỮ LIỆU TRUY VẤN VÀO BẢNG:

Nội dung bảng customers:

	cust_id	cust_name	cust_address	cust-city
1	1000000001	Viilage Toys	200 Maple Lane	Detroit
2	1000000002	Kids Place	333 South Lake Drive	Columbus
3	1000000003	Fun4All	1 Sunny Place	Muncie
4	1000000004	Fun4All	829 Riverside Drive	Phoenix
5	1000000005	The Toy Store	4545 53rd Street	Chicago

cust_state	cust_zip	cust_country	cust_contact	cust_email
MI	44444	USA	John Smith	sales@villagetoys.com
OH	43333	USA	Michelle Green	
IN	42222	USA	Jim Jones	jjones@fun4all.com
AZ	88888	USA	Denise L. Stephens	dstephens@fun4all.com
IL	54545	USA	Kim Howard	

Nội dung bảng custnew:

	cust_id	cust_name	cust_address	cust-city
1	1000000007	Lê Trần Diễm Quý	12/5,Lý Thường Kiệt,Q.Tân Bình	HỒ CHÍ MINH
2	1000000008	Nguyễn Hữu Nghĩa	172,ĐINH TIÊN HOÀNG,Q1	HỒ CHÍ MINH
3	1000000008	NGUYỄN THỊ HỒNG	172,ĐINH TIÊN HOÀNG,Q1	HỒ CHÍ MINH
4	1000000009	NGUYỄN CÔNG TUẤN	121,LÝ THƯỜNG KIỆT,Q.TÂN BÌNH	HỒ CHÍ MINH

cust_state	cust_zip	cust_country	cust_contact	cust_email
HCM	55555	VIỆT NAM	CÔNG TUẤN	letrandiemquy@yahoo.com
HCM	44444	VIỆT NAM		
HCM	44444	VIỆT NAM		
HCM	66666	VIỆT NAM		

```
INSERT INTO CUSTOMERS
(
        cust_id,
        cust_name,
        cust_address,
        cust_city,
        cust_state,
        cust_zip,
        cust_country
)
SELECT
        cust_id,
        cust_name,
        cust_address,
        cust_city,
        cust_state,
        cust_zip,
        cust_country
FROM CUSTNEW;
```

Trong ví dụ trên bạn đã chèn dữ liệu do truy xuất từ bảng CUSTNEW vào bảng CUSTOMERS.

```
SELECT *
FROM CUSTOMERS;
```

Màn hình xuất ra:

	cust_id	cust_name	cust_address	cust-city
1	1000000001	Viilage Toys	200 Maple Lane	Detroit
2	1000000002	Kids Place	333 South Lake Drive	Columbus
3	1000000003	Fun4All	1 Sunny Place	Muncie
4	1000000004	Fun4All	829 Riverside Drive	Phoenix
5	1000000005	The Toy Store	4545 53rd Street	Chicago
6	1000000007	Lê Trần Diễm Quý	12/5,Lý Thường Kiệt,Q.Tân Bình	HỒ CHÍ MINH
7	1000000008	Nguyễn Hữu Nghĩa	172,ĐINH TIÊN HOÀNG,Q1	HỒ CHÍ MINH
8	1000000008	NGUYỄN THỊ HỒNG	172,ĐINH TIÊN HOÀNG,Q1	HỒ CHÍ MINH
9	1000000009	NGUYỄN CÔNG TUẤN	121,LÝ THƯỜNG KIỆT,Q.TÂN BÌNH	HỒ CHÍ MINH

	cust_stat	cust_zi	cust_countr	cust_contact	cust_email
1	MI	44444	USA	John Smith	sales@villagetoys.com
2	OH	43333	USA	Michelle Green	
3	IN	42222	USA	Jim Jones	jjones@fun4all.com
4	AZ	88888	USA	Denise L. Stephens	dstephens@fun4all.com
5	IL	54545	USA	Kim Howard	
6	HCM	55555	VIỆT NAM	NULL	NULL
7	HCM	44444	VIỆT NAM	NULL	NULL
8	HCM	44444	VIỆT NAM	NULL	NULL
9	HCM	66666	VIỆT NAM	NULL	NULL

4. CHÉP TỪ MỘT BẢNG TỚI BẢNG KHÁC:

Giả sử bạn cần chép dữ liệu từ bảng customers sang một bảng khác có tên custcopy (bảng mới) chẳng hạn, cú pháp:

```
SELECT *
INTO CUSTCOPY
FROM CUSTOMERS;
```

Nếu bạn đang dùng trình cơ sở dữ liệu Oracle thì cú pháp như sau:

```
CREATE TABLE CUSTCOPY AS
SELECT *
FROM CUSTOMERS;
```

BÀI 16

CẬP NHẬT VÀ XÓA DỮ LIỆU

Trong bài học này bạn sẽ học cách sử dụng câu lệnh UPDATE (cập nhật) và DELETE (xóa) để thực hiện việc chỉnh sửa bảng dữ liệu.

1. CẬP NHẬT DỮ LIỆU:

Để cập nhật dữ liệu ta sử dụng câu lệnh UPDATE, có hai cách thực hiện như sau:

- Cập nhật từng dòng của bảng.
- Cập nhật tất cả các dòng có trong bảng.

Việc thực hiện tương đối đơn giản, theo 3 bước sau:

- Xác định bảng cần được cập nhật.
- Tên các cột và dữ liệu cần cập nhật tương ứng.
- Bộ lọc điều kiện để xác nhận các cột được cập nhật.

Ví dụ bạn có một khách hàng (customer) có cust_id là 100000005 trong cột cust_email chưa có địa chỉ email, bây giờ nhiệm vụ của bạn là thêm vào đó địa chỉ email:

```
UPDATE CUSTOMERS
SET CUST_EMAIL='kim@thetoystore.com'
WHERE CUST_ID='1000000005';
```

Lúc này bạn sẽ thấy trong dòng (record) 5 xuất hiện địa chỉ email của cột cust_email. Bạn nên lưu ý mệnh đề lọc điều kiện 'WHERE' trong ví dụ trên là rất quan trọng.

Một cách khác bạn có thể cập nhật dữ liệu cho nhiều cột trong bảng ghi:

```
UPDATE CUSTOMERS
SET   CUST_CONTACT='NGUYEN CONG TUAN',
      CUST_EMAIL='congtuannguyendhkt@hotmail.com'
WHERE CUST_ID='1000000007';
```

Lúc này bảng ghi số 7 trong bảng customers sẽ được cập nhật hai cột là cust_contact và cust_email.

Tuy nhiên nếu bạn muốn xóa đi giá trị của một cột nào đó có trong bảng ghi thứ n chẳng hạn bạn có thể gán cho nó một giá trị là NULL:

```
UPDATE CUSTOMERS
SET   CUST_EMAIL=NULL
WHERE CUST_ID='1000000007';
```

Lúc này cột cust_email có trong record thứ 7 sẽ mang giá trị NULL.

2. XÓA DỮ LIỆU:

Để xóa (remove) dữ liệu từ một bảng bạn có thể dùng câu lệnh DELETE với 2 cách:

- Xóa các dòng được chỉ định từ bảng.
- Xóa tất cả các dòng có trong bảng.

Ví dụ bạn cần xóa record thứ 6 có trong bảng customers chẳng hạn thì cú pháp:

```
DELETE FROM CUSTOMERS
WHERE CUST_ID='1000000006';
```

Ngoài ra nếu bạn muốn xóa tất cả các dòng trong bảng một cách nhanh chóng bạn không nên dùng câu lệnh DELETE mà bạn hãy sử dụng câu lệnh TRUNCATE TABLE <tên bảng>;

Ví dụ bạn cần xóa tất cả các dòng có trong bảng custnew, cú pháp:

```
TRUNCATE TABLE CUSTNEW;
```

Lưu ý:

Bạn chỉ có thể xóa tất cả các record có trong table, bạn sẽ không thể xóa chính bảng đó.

BÀI 17

TẠO VÀ CHỈNH SỬA BẢNG

Trong bài học này bạn sẽ học căn bản về cách tạo, sự thay đổi và sửa chữa bảng.

1. TẠO BẢNG:

Có hai cách tạo ra bảng:

- Hầu hết các trình cơ sở dữ liệu có hỗ trợ DBMS' đều có các công cụ (tools) để tạo và quản lí bảng.
- Bảng cũng có thể được tạo trực tiếp bằng tay (thủ công).

Để tạo bảng ta dùng câu lệnh CREATE TABLE <tên bảng>, với một số lưu ý sau:

- Bạn phải có tên của bảng muốn tạo sau từ khóa CREATE TABLE.
- Tên của các cột có trong bảng và các kiểu định dạng của chúng phải có.
- Một số cấu trúc DBMS' nếu có.

Ví dụ bạn cần tạo một bảng tiền lương của nhân viên trong công ty chẳng hạn:

```
CREATE TABLE SALARYS
(
    company_id          char(10)          not null,
    company_name        char(50)          not null,
    worker_id           char(10)          not null,
    worker_name         char(50)          not null,
    worker_salary       decimal(8,2)      not null,
```

```
    worker_desc          varchar(1000)
);
```

Như vậy ta đã tạo được một bảng lương SALARYS có 6 cột tương ứng với mỗi cột có một kiểu định dạng riêng.

2. LÀM VIỆC VỚI GIÁ TRỊ RỖNG (NULL):

Như các bạn NULL là giá trị rỗng, trong SQL các cột thường buộc phải có giá trị (NOT NULL) còn trong khi đó các dòng (record) không bắt buộc tức là có thể không mang giá trị (NULL value).

Ví dụ ta sẽ tạo ra một bảng mà trong đó sẽ có các cột **not null** và cả các cột null. Bạn lưu ý nếu bạn chọn giá trị của cột là **not null** mà bạn không điền dữ liệu vào cột đó nó sẽ là không hợp lệ.

```
CREATE TABLE STUDENTS
(
    student_id          char(10)          not null,
    student_name        char(50)          not null,
    university_id       char(10)          not null,
    university_name     char(50)          not null,
    student_desc        varchar(1000),
    university_desc     varchar(1000)
);
```

Trong ví dụ trên bạn đã tạo được một bảng sinh viên với 6 cột trong đó có 4 cột đầu **not null** tức là bạn phải điền dữ liệu vào đó còn 2 cột sau **null** tức là rỗng.

3. GIÁ TRỊ MẶC ĐỊNH (DEFAULT):

SQL cho phép bạn gán giá trị mặc định cho trước vào cột nếu như nó bị bỏ trống (empty).

Ví dụ:

```
CREATE TABLE GOLDS
(
        gold_id         char(10)        not null,
        gold_name       char(50)        not null,
        quantity        integer         not null        default 1,
        gold_price      decimal(8,2)    not null
);
```

Trong ví dụ trên bạn thấy đã tạo ra được bảng GOLDS với 4 cột trong đó cột thứ 3 sử dụng giá trị mặc định (default), giá trị này mang ý nghĩa là khi cột số lượng (quantity) trống thì tự động nó sẽ cập nhật vào đó giá trị là 1.

Hoặc:

```
CREATE TABLE ORDERS
(
ORDER_NUM    INTEGER    NOT NULL,
ORDER_DATE   DATETIME   NOT NULL DEFAULT  SYSDATE,
CUST_ID      CHAR(10)   NOT NULL
);
```

Trong ví dụ trên bạn thấy nếu ta không điền giá trị ngày giờ của cột **order_date** thì nó sẽ tự động cập nhật vào đó ngày giờ của hệ thống. Hoặc nếu như hàm ngày giờ của SQL không có sẵn bạn có thể sử dụng hàm GETDATE() để thực hiện việc trên cho ra kết quả tương tự.

```
CREATE TABLE ORDERS
(
ORDER_NUM    INTEGER    NOT NULL,
ORDER_DATE   DATETIME   NOT NULL DEFAULT  GETDATE(),
CUST_ID      CHAR(10)   NOT NULL
);
```

Cuối cùng, trong phiên bản Microsoft Access bạn có thể sử dụng hàm NOW() để lấy ngày giờ hệ thống.

```
CREATE TABLE ORDERS
(
ORDER_NUM    INTEGER     NOT NULL,
ORDER_DATE   DATETIME    NOT NULL DEFAULT  NOW (),
CUST_ID      CHAR(10)    NOT NULL
);
```

4. CẬP NHẬT BẢNG:

Để Update bảng ta dùng câu lệnh ALTER TABLE để thực hiện, song có một số lưu ý sau:

- Tên của bảng phải đứng sau câu lệnh ALTER TABLE, tức là bảng phải tồn tại.
- Những giá trị được cập nhật.

Ví dụ bạn cần cập nhật thêm vào bảng VENDORS một số cột sau:

```
VEND_PHONE          CHAR(20)
VEND_COMPUTER       CHAR(50)
```

Cú pháp:

```
ALTER TABLE VENDORS
ADD
    VEND_PHONE      CHAR(20),
    VEND_COMPUTER   CHAR(50);
```

Nếu bạn muốn bỏ cột nào thì bạn sẽ dùng câu lệnh DROP COLUMN <tên cột>, giả sử bạn muốn bỏ cột vend_phone vừa tạo.

Cú pháp:

```
ALTER TABLE VENDORS
DROP COLUMN VEND_PHONE;
```

Một số lưu ý:

- Sẽ không cập nhật chồng lên dữ liệu đã có sẵn.
- Tất cả DBMS' cho phép bạn thêm cột tới một bảng tồn tại song có một vài giới hạn về kiểu dữ liệu được thêm (chẳng hạn như khi sử dụng NULL hay DEFAULT).
- Nhiều DBMS' sẽ không cho bạn gỡ bỏ hay thay đổi cột của bảng.
- Hầu hết DBMS' cho phép bạn đổi tên cột.

Những cú pháp thực hiện việc cập nhật khác các bạn sử dụng cú pháp tương tự.

5. XÓA BẢNG:

Để xóa bảng đơn giản bạn chỉ cần dùng câu lệnh DROP TABLE <tên bảng>. Ví dụ bạn cần xóa bảng custnew đi bạn chỉ cần thực hiện như sau:

```
DROP TABLE CUSTNEW;
```

6. ĐỔI TÊN BẢNG:

Bạn có thể sử dụng lệnh SP_RENAME để đổi tên bảng.

BÀI 18

SỬ DỤNG VIEWS

Trong bài học này bạn sẽ được học rõ VIEWS, cách nó hoạt động và chúng ta sử dụng VIEWS như thế nào.

1. HIỂU RÕ VỀ VIEWS:

Views là một bảng ảo, nó không giống như những bảng chứa dữ liệu khác, **views** chỉ đơn giản là bao gồm các câu lệnh truy vấn và nó sẽ thực hiện truy vấn dữ liệu khi **views** hoạt động.

Ví dụ:

```
SELECT CUST_NAME,CUST_CONTACT
FROM CUSTOMERS,ORDERS,ORDERITEMS
WHERE CUSTOMERS.CUST_ID=ORDERS.CUST_ID
AND ORDERITEMS.ORDER_NUM=ORDERS.ORDER_NUM
AND PROD_ID='RGAN01';
```

Trong ví dụ trên bạn đã sử dụng câu lệnh truy vấn để truy xuất những khách hàng đã đặt hàng mua sản phẩm có hóa đơn trong bảng **orders**. Như bạn đã biết trong ví dụ trên ta đã sử dụng liên kết nhiều bảng lại để truy xuất dữ liệu với nhau.

Bây giờ bạn hãy tưởng tượng rằng ta có thể gói (wrap) toàn bộ các câu lệnh truy vấn vào trong một bảng ảo (virtual table) được gọi là Productcustomers chẳng hạn. Vì vậy sau này thay vì để làm việc trên với rất nhiều câu lệnh rườm rà bạn có thể đơn

giản hơn bằng cách đánh câu lệnh sau nhưng cho ra cùng kết quả:

```
SELECT CUST_NAME,CUST_CONTACT
FROM PRODUCTCUSTOMERS
WHERE PROD_ID='RGAN01';
```

Trong đó productcustomers là một **view** hay là một **virtual table**.

2. MỘT SỐ LÍ DO ĐỂ BẠN SỬ DỤNG VIEWS:

- Dùng lại các câu lệnh SQL.
- Đơn giản các câu lệnh SQL vốn phức tạp. Chúng ta có thể dùng các **views** mà không cần biết chi tiết nội dung bên trong của nó.
- Đưa ra những bộ phận của bảng thay vì toàn bộ bảng.
- Bảo mật dữ liệu. Người dùng có thể cho xem chỉ một phần của bảng thay vì cho xem toàn bộ bảng.
- Thay đổi được định dạng của dữ liệu và các giá trị hiện thời. **Views** có thể đưa ra các định dạng dữ liệu khác nhau từ bảng.

3. MỘT SỐ NGUYÊN TẮC VÀ GIỚI HẠN CỦA VIEWS:

- Giống như bảng, views chỉ được đặt duy nhất một tên (tên không được trùng với tên các bảng khác và các **views** đã có sẵn).
- Số Views tạo ra sẽ không bị giới hạn.
- Để tạo được **views** bạn phải có quyền truy nhập vào bảo mật và điều này được cấp quyền bởi Administrator.
- **Views** phải được cấu trúc theo nhóm mà theo đó các **views** khác có thể truy xuất dữ liệu với nhau.
- Bạn có thể được sử dụng mệnh đề ORDER BY vào câu lệnh truy vấn của **views**.

- Views không thể được đặt các chỉ mục.

4. TẠO VIEWS:

Bạn tạo **views** bằng cách sử dụng câu lệnh CREATE VIEW. Bạn nên nhớ **views** mà bạn tạo ra phải hoàn toàn mới, chưa tồn tại.

a) Sử dụng views để đơn giản các liên kết phức tạp:

Một trong số những ưu điểm của views là bỏ qua bớt một số cấu trúc SQL phức tạp và điều này bao gồm cả các liên kết bảng.

Ví dụ:

```
CREATE VIEW PRODUCTCUSTOMERS AS
SELECT CUST_NAME,CUST_CONTACT,PROD_ID
FROM CUSTOMERS,ORDERS,ORDERITEMS
WHERE CUSTOMERS.CUST_ID=ORDERS.CUST_ID
AND ORDERITEMS.ORDER_NUM=ORDERS.ORDER_NUM
```

Trong ví dụ trên bạn đã tạo ra được một view là productcustomers, nó đã liên kết ba bảng để đưa ra danh sách của tất cả các khách hàng đã đặt hàng.

Lưu ý:

Trong câu lệnh CREATE VIEW kết thúc sẽ không có dấu chấm phẩy (;).

Bây giờ nếu bạn sử dụng câu lệnh:

```
SELECT *
FROM PRODUCTCUSTOMERS;
```

Lúc này nó sẽ truy xuất tất cả các dữ liệu có trong view productcustomers cho bạn xem:

	CUST_NAME	CUST_CONTACT	PROD_ID
1	Village Toys	John Smith	BR01
2	Village Toys	John Smith	BR03
3	Fun4All	Jim Jones	BR01
4	Fun4All	Jim Jones	BR02
5	Fun4All	Jim Jones	BR03
6	Fun4All	Denise L. Stephens	BR03
7	Fun4All	Denise L. Stephens	BNBG01
8	Fun4All	Denise L. Stephens	BNBG02
9	Fun4All	Denise L. Stephens	BNBG03
10	Fun4All	Denise L. Stephens	RGAN01
11	The Toy Store	Kim Howard	RGAN01
12	The Toy Store	Kim Howard	BR03
13	The Toy Store	Kim Howard	BNBG01
14	The Toy Store	Kim Howard	BNBG02
15	The Toy Store	Kim Howard	BNBG03
16	Village Toys	John Smith	BNBG01
17	Village Toys	John Smith	BNBG02
18	Village Toys	John Smith	BNBG03

Bây giờ bạn có thể truy xuất dữ liệu từ view productcustomers:

```
SELECT CUST_NAME,CUST_CONTACT
FROM PRODUCTCUSTOMERS
WHERE PROD_ID='RGAN01';
```

Màn hình xuất ra:

	CUST_NAME	CUST_CONTACT
1	Fun4All	Denise L. Stephens
2	The Toy Store	Kim Howard

b) Sử dụng views để định dạng lại dữ liệu truy xuất:

Bây giờ bạn cần trả về thông tin địa chỉ cho tất cả các khách hàng có trong bảng customers:

```
SELECT
    CUST_NAME,
    CUST_ADDRESS,
    CUST_CITY,
    CUST_STATE,
    CUST_ZIP,
    CUST_COUNTRY
    FROM CUSTOMERS;
```

Màn hình xuất ra:

	CUST_NAME	CUST_ADDRESS
1	Viilage Toys	200 Maple Lane
2	Kids Place	333 South Lake Drive
3	Fun4All	1 Sunny Place
4	Fun4All	829 Riverside Drive
5	The Toy Store	4545 53rd Street
6	Lê Trần Diễm Quý	12/5,Lý Thường Kiệt,Q.Tân Bình
7	Nguyễn Hữu Nghĩa	172,ĐINH TIÊN HOÀNG,Q1
8	NGUYỄN THỊ HỒNG	172,ĐINH TIÊN HOÀNG,Q1
9	NGUYỄN CÔNG TUẤN	121,LÝ THƯỜNG KIỆT,Q.TÂN BÌNH ...

CUST_CITY	CUST_STATE	CUST_ZIP	CUST_COUNTRY
Detroit	MI	44444	USA
Columbus	OH	43333	USA
Muncie	IN	42222	USA
Phoenix	AZ	88888	USA
Chicago	IL	54545	USA
HỒ CHÍ MINH	HCM	55555	VIỆT NAM
HỒ CHÍ MINH	HCM	44444	VIỆT NAM
HỒ CHÍ MINH	HCM	44444	VIỆT NAM
HỒ CHÍ MINH	HCM	66666	VIỆT NAM

Ví dụ bây giờ do nhu cầu mailing bạn chỉ có thể in bảng customers ra làm bốn cột thô. Một bạn dành cho tên khách hàng và ba cột còn lại bạn bố trí cho nó là ba cột địa chỉ được phân bố như sau:

```
SELECT CUST_NAME,
CUST_ADDRESS AS CUST_ADDRESS1,
RTRIM(CUST_CITY)+','+RTRIM(CUST_STATE)+'
'+RTRIM(CUST_ZIP)
AS CUST_ADDRESS2,
CUST_COUNTRY AS CUST_ADDRESS3
FROM CUSTOMERS;
```

Màn hình xuất ra:

	CUST_NAME	CUST_ADDRESS1	CUST_ADDRESS2	CUST_ADDRESS3
1	Viilage Toys	200 Maple Lane	Detroit,MI44444	USA
2	Kids Place	333 South Lake Drive	Columbus,OH43333	USA
3	Fun4All	1 Sunny Place	Muncie,IN42222	USA
4	Fun4All	829 Riverside Drive	Phoenix,AZ88888	USA
5	The Toy Store	4545 53rd Street	Chicago,IL54545	USA
6	Lê Trần Diễm Quý	12/5,Lý Thường Kiệt,Q.Tân...	HỒ CHÍ MINH,HCM55555	VIỆT NAM
7	Nguyễn Hữu Nghĩa	172,ĐINH TIÊN HOÀNG,...	HỒ CHÍ MINH,HCM44444	VIỆT NAM
8	NGUYỄN THỊ HỒNG ...	172,ĐINH TIÊN HOÀNG,..	HỒ CHÍ MINH,HCM44444	VIỆT NAM
9	NGUYỄN CÔNG TUẤN	121,LÝ THƯỜNG KIỆT,Q. ...	HỒ CHÍ MINH,HCM66666	VIỆT NAM

Bây giờ bạn hãy tạo một view mang tên Customermailinglables như sau:

```
CREATE VIEW Customermailinglables AS
SELECT CUST_NAME,
                CUST_ADDRESS AS CUST_ADDRESS1,
                RTRIM(CUST_CITY)+','+RTRIM(CUST_ST
ATE)+''+RTRIM(CUST_ZIP) AS CUST_ADDRESS2,
                CUST_COUNTRY AS CUST_ADDRESS3
FROM CUSTOMERS
```

Bây giờ bạn đã có **view Customermailinglables**, bạn có thể truy cập dữ liệu từ **view** này như sau:

```
SELECT *
FROM Customermailinglables;
```

Nếu bạn đang sử dụng phiên bản Oracle thì cú pháp như sau:

```
SELECT CUST_NAME,
                CUST_ADDRESS AS CUST_ADDRESS1,
                RTRIM(CUST_CITY)||','||RTRIM(CUST_
                STATE)||' '||RTRIM(CUST_ZIP)
                AS CUST_ADDRESS2,
                CUST_COUNTRY AS CUST_ADDRESS3
FROM CUSTOMERS;
```

c) Sử dụng views để lọc dữ liệu ta không cần:

Views cũng hữu ích cho việc dùng với mệnh đề WHERE lọc ra các dữ liệu ta không cần. Ví dụ bạn muốn lọc bảng ảo Customermailinglables để từ đó loại ra các đối tượng rỗng (NULL).

Cách thực hiện:

```
CREATE VIEW Customermailinglables01 AS
SELECT      CUST_NAME,
            CUST_ADDRESS AS CUST_ADDRESS1,
        RTRIM(CUST_CITY)+','+RTRIM(CUST_STATE)+''
+RTRIM(CUST_ZIP) AS CUST_ADDRESS2,
            CUST_COUNTRY AS CUST_ADDRESS3
  FROM CUSTOMERS
  WHERE     NOT       CUST_ADDRESS IS NULL
            AND NOT   CUST_CITY IS NULL
            AND NOT   CUST_STATE IS NULL
            AND NOT   CUST_ZIP IS NULL
            AND NOT   CUST_COUNTRY IS NULL
            AND NOT   CUST_EMAIL IS NULL
```

Ta tiến hành truy xuất view Customermailinglables01 thu được:

```
SELECT *
FROM Customermailinglables01;
```

Màn hình xuất ra:

	CUST_NAME	CUST_ADDRESS1	CUST_ADDRESS2	CUST_ADDRESS
1	Viilage Toys	200 Maple Lane	Detroit,MI44444	USA
2	Kids Place	333 South Lake Drive	Columbus,OH43333	USA
3	Fun4All	1 Sunny Place	Muncie,IN42222	USA
4	Fun4All	829 Riverside Drive	Phoenix,AZ88888	USA
5	The Toy Store	4545 53rd Street	Chicago,IL54545	USA

Trong ví trên ta thấy view Customermailinglables01 đã lọc đi năm dòng 6, 7, 8, 9 có cust_email là null.

d) Sử dụng views với trường tính toán:

Ví dụ bạn có một calculated field như sau:

```
SELECT PROD_ID,
        QUANTITY,
        ITEM_PRICE,
        QUANTITY*ITEM_PRICE AS EXPANDED_PRICE
FROM ORDERITEMS
WHERE ORDER_NUM=20008;
```

Màn hình xuất ra:

	PROD_ID	QUANTITY	ITEM_PRICE	EXPANDED_PRICE
1	RGAN01	5	4.99	24.95
2	BR03	5	11.99	59.95
3	BNBG01	10	3.49	34.90
4	BNBG02	10	3.49	34.90
5	BNBG03	10	3.49	34.90

Bây giờ bạn hãy tạo một **view** để thay thế:

```
CREATE VIEW ORDERITEMSEXPANDED AS
SELECT
        ORDER_NUM,
        PROD_ID,
        QUANTITY,
        ITEM_PRICE,
        QUANTITY*ITEM_PRICE AS EXPANDED_PRICE
FROM ORDERITEMS
```

Ta truy xuất dữ liệu từ view ORDERITEMSEXPANDED sẽ cho kết quả:

```
SELECT *
FROM ORDERITEMSEXPANDED
WHERE ORDER_NUM='20008';
```

Màn hình xuất ra:

	ORDER_NUM	PROD_ID	QUANTITY	ITEM_PRICE	EXPANDED_PRICE
1	20008	RGAN01	4.99	24.95	5
2	20008	BR03	11.99	59.95	5
3	20008	BNBG01	3.49	34.90	10
4	20008	BNBG02	3.49	34.90	10
5	20008	BNBG03	3.49	34.90	10

BÀI 19

LÀM VIỆC VỚI THỦ TỤC LƯU TRỮ

Trong bài học này bạn sẽ được học căn bản về thủ tục lưu trữ: khái niệm, cách sử dụng,...

Hiểu rõ về thủ tục lưu trữ (Stored Procedures):

Thực ra Stored Procedures là một tập hợp các câu lệnh select được đóng gói theo một kiểu nhất định để lần sau bạn có thể thực thi.

Stored Procedures thực thi:

```
EXECUTE ADDNEWPRODUCT
(
    'TS01',
    'Stuffed Eiffel Tower',
    6.49,
    'Push stuffed toy with the text La Tour Eiffel in red white and blue'
)
```

Như bạn thấy ADDNEWPRODUCT là một Stored Procedure được thực thi. Nó thêm một sản phẩm mới vào trong bảng sản phẩm. AddnewProduct đưa ra 4 tham số: vend_id (là primary key của bảng vendors), prod_name, prod_price và prod_desc. Bốn tham số này được xem như 4 biến số trong Stored procedure (định nghĩa nên stored procedure). Nó đã thêm một dòng mới vào bảng sản phẩm và gán các biến số này cho các cột tương ứng.

Tạo **Stored Procedures**:

Bây giờ bạn sẽ tạo một stored procedures đếm số khách hàng có nhận email:

```
CREATE PROCEDURE MAILINGLISTCOUNT
AS
DECLARE @cnt INTEGER
SELECT COUNT(*) AS cnt
FROM CUSTOMERS
WHERE NOT CUST_EMAIL IS NULL
RETURN @cnt;
```

Trong ví dụ trên stored procedures không chứa bất cứ tham số nào. Ứng dụng được gọi sẽ truy xuất giá trị bằng cách sử dụng mã của SQL server trả về. @cnt là một biến cục bộ và nó được khai báo bởi câu lệnh DECLARE (tất cả các biến trong SQL server được đặt tên với bắt đầu là @). Biến này sau đó sẽ được sử dụng bởi câu lệnh SELECT, nó chứa các giá trị trong hàm Count(*) trả về. Cuối cùng thì câu lệnh Return được sử dụng để đưa ra nội dung của biến cnt, return @cnt.

Bạn có thể xem cách thực hiện như sau:

```
EXECUTE MAILINGLISTCOUNT
```

Màn hình xuất ra:

	nt
1	5

Nếu bạn sử dụng phiên bản Oracle thì cú pháp sẽ là:

```
CREATE PROCEDURE MAILINGLISTCOUNT
(listcount out number)
IS
BEGIN
```

```
SELECT *
FROM CUSTOMERS
WHERE NOT CUST_EMAIL IS NULL
LISTCOUNT:=SQL%ROWCOUNT;
END;
```

Tiếp theo là một ví dụ khác, lần này ta sẽ chèn thêm một hóa đơn mới vào trong bảng orders:

```
CREATE PROCEDURE NewOder
@ cust _id CHAR(10)
AS
/*Khai báo biến cho số hóa đơn*/
DECLARE @order_num INTEGER
/*Trả về số hóa đơn cao nhất ở thời điểm hiện tại*/
SELECT @order_num=MAX(order_num)
FROM ORDERS
/*Xác định rõ hóa đơn tiếp theo*/
SELECT @order_num=@order_num+1
/*Chèn một hóa đơn mới*/
INSERT INTO Orders
        (order_num,
        order_date,
        cust_id)
VALUES
                (@order_num,
                GETDATE( ),
                @cust _id)
        /*Trả về số hóa đơn*/
RETURN @order_num;
```

Hoặc:

```
CREATE PROCEDURE NEWORDERS
@ CUST_ID CHAR(10)
AS
/*Chèn thêm một hóa đơn mới*/
INSERT INTO Orders
        (
        CUST_ID
        )
VALUES
        (
        @CUST_ID
        )
/*Trả về số hóa đơn*/
SELECT @ @IDENTITY AS ORDER_NUM;
```

BÀI 20

KIỂM SOÁT VIỆC XỬ LÝ GIAO TÁC

Trong bài học này bạn sẽ học các Transactions (xử lí) và cũng như cách sử dụng các câu lệnh COMMIT và ROLLBACK để kiểm soát việc xử lí các giao tác.

1. HIỂU VỀ XỬ LÍ GIAO TÁC (Transaction Processing):

Thực ra xử lí giao tác được sử dụng để duy trì tính toàn vẹn của dữ liệu bằng cách xử lí các khối dữ liệu của các câu lệnh thực thi hoàn thành hay chưa hoàn thành. Bây giờ ta lấy một ví dụ minh hoạ cho điều này:

Các hóa đơn thì được đặt trong hai bảng là orders và orderitems, hai bảng này có mối quan hệ với nhau qua IDs hay còn gọi là khóa chính. Ngoài ra hai bảng này còn quan hệ với các bảng khác như customers, products. Thao tác thêm một hóa đơn tới hệ thống được cho phép như sau:

- Kiểm tra xem khách hàng có trong cơ sở dữ liệu chưa, nếu chưa hãy thêm vào.
- Truy xuất IDs của khách hàng.
- Thêm một dòng đến bảng orders thông qua IDs của khách hàng đó.
- Truy xuất IDs của khách hàng mới qua bảng orders.

- Thêm một dòng tới bảng orderitems cho mỗi item được

đặt hàng thông qua bảng orders bằng cách truy xuất IDs (cùng với bảng products bởi IDs).

Bây giờ bạn hãy nghĩ rằng dữ liệu bị lỗi (tràn bộ nhớ đĩa cứng, giới hạn của bảo mật, khóa bảng,...) từ đó không cho dữ liệu truyền đi. Vậy lúc này điều gì sẽ xảy ra với dữ liệu của chúng ta? Chúng có thể bị mất và làm hỏng bảng dữ liệu của ta, vậy giải pháp là gì? Lúc này bạn có thể sử dụng Transaction Processing để giải quyết. Công việc của nó rất đơn giản là làm cho dữ liệu truyền đi một cách tốt đẹp (COMMITTED) ngoài ra trên đường truyền bị lỗi nó sẽ lập tức đưa dữ liệu về một cách an toàn (ROLLBACKED).

Sau đây là toàn bộ qui trình xử lí của transaction processing:

- Kiểm tra xem dữ liệu về khách hàng tồn tại, nếu không thì thêm vào.
- Chuyển thông tin khách hàng.
- Truy xuất IDs của khách hàng.
- Thêm một dòng tới bảng orders.
- Nếu bị lỗi trong khi chuyển record tới bảng orders thì chuyển về (rollback).
- Truy xuất ID của hóa đơn mới được gán trong bảng orders.
- Thêm dòng tới bảng orderitems cho mỗi hóa đơn đã đặt.
- Nếu lỗi trong khi chuyển dòng tới bảng orderitems thì trả về bảng orderitems dòng đã được thêm và dòng trong bảng orders.

Trong khi bạn làm việc với Transactions hay Transaction processing, có một số từ khóa bạn nên biết:

TRANSACTION: một khối (block) của câu lệnh SQL.

ROLLBACK: tác vụ trả về của câu lệnh SQL.

COMMIT: thêm (không lưu) câu lệnh SQL tới bảng cơ sở dữ liệu.

SAVEPOINT: một thư mục tạm để khi ROLLBACK trả dữ liệu về.

Lưu ý:

Bạn chỉ có thể rollback các dữ liệu khi sử dụng các câu lệnh INSERT, UPDATE hay DELETE. Bạn không thể rollback các câu lệnh SELECT, CREATE hay DROP.

2. ĐIỀU KHIỂN CÁC GIAO TÁC:

Một số DBMS' yêu cầu bạn phải đánh dấu điểm khởi đầu và điểm kết thúc của khối giao tác.

```
BEGIN TRANSACTION
...
END TRANSACTION
```

a) Sử dụng ROLLBACK:

```
DELETE FROM CUSTNEW;
ROLLBACK;
```

Trong ví dụ trên ta đã xóa các dữ liệu của custnew, sau đó undo trở lại nhờ rollback.

b) Sử dụng COMMIT:

Ví dụ bạn thực hiện một transaction là xóa đi order_num=12345 của bảng orderitems và order_num=12345 của bảng orders:

```
BEGIN TRANSACTION
DELETE      ORDERITEMS
WHERE       ORDER_NUM=1000000008;
DELETE      ORDERS
WHERE       ORDER_NUM=1000000008;
COMMIT TRANSACTION;
```

Trong Oracle bạn có thể thực hiện như sau:

```
DELETE      ORDERITEMS
WHERE       ORDER_NUM=1000000008;
DELETE      ORDERS
WHERE       ORDER_NUM=1000000008;
COMMIT;
```

c) Sử dụng SavePoints:

Câu lệnh như sau:

```
SAVE TRANSACTION DELETE1;
```

Còn trong Oracle thì:

```
SAVEPOINT DELETE1;
```

Bây giờ bạn có thể rollback dữ liệu về từ thư mục tạm (placeholders):

```
ROOLBACK TRANSACTION DELETE1;
```

Trong Oracle thì câu lệnh:

```
ROOLBACK TO DELETE1;
```

Sau đây là một ví dụ tổng hợp cho các tác vụ trên:

```
BEGIN TRANSACTION
INSERT INTO CUSTOMERS
(
CUST_ID,
CUST_NAME
)
VALUES
(
'1000000010',
'TOYS EMPORIUM'
);
SAVE TRANSACTION STARTORDER;
INSERT INTO ORDERS
(
```

```
ORDER_NUM,
ORDER_DATE,
CUST_ID
)
VALUES
(
20100,
GETDATE(),
'1000000010'
);
IF @@ERROR<>0 ROLLBACK TRANSACTION STARTORDER;
INSERT INTO ORDERITEMS
(
ORDER_NUM,
ORDER_ITEM,
PROD_ID,
QUANTITY,
ITEM_PRICE
)
VALUES
(
20010,
1,
'BR01',
100,
5.49
);
IF @@ERROR <>0 ROLLBACK TRANSACTION STARTORDER;
INSERT INTO ORDERITEMS
(
```

```
ORDER_NUM,
ORDER_ITEM,
PROD_ID,
QUANTITY,
ITEM_PRICE
)
VALUES
(
20010,
2,
'BR03',
100,
10.99
);
IF @@ERROR<>0 ROLLBACK TRANSACTION STARTORDER;
COMMIT TRANSACTION
```

Đây là tập hợp của 4 câu lệnh INSERT đính kèm với các transaction block. Một Savepoint (placeholder) được tạo ra sau câu lệnh insert đầu tiên, vì vậy nếu có bất cứ câu truy vấn nào của câu lệnh insert bị lỗi thì transaction sẽ rollback về savepoint. Trong SQL nếu có một biến được gán tên là @@ERROR thì có nghĩa là toán tử thực hiện thành công. Vì vậy nếu biến @@ERROR trả về giá trị là 0 tức là đã bị lỗi, lập tức transaction rollback đưa về placeholders. Còn nếu toàn bộ transaction thành công thì câu lệnh COMMIT được thực thi để lưu dữ liệu lại.

BÀI 21

SỬ DỤNG CON TRỎ

Trong bài học này bạn sẽ được học thế nào là con trỏ (cursor) và cách sử dụng chúng.

Một con trỏ là một câu truy vấn dữ liệu được chứa trong DBMS server, nó thực chất không phải là một câu lệnh SELECT nhưng kết quả truy xuất được thực hiện bởi câu lệnh đó. Con trỏ có thể hiểu là sự lưu trữ, là các ứng dụng mà ta có thể xem, truy xuất dữ liệu khi ta cần.

1. LÀM VIỆC VỚI CON TRỎ:

Sử dụng con trỏ thực hiện theo nhiều bước khác nhau:

- Trước khi một con trỏ được sử dụng nó phải được khai báo (định nghĩa). Điều này không phải là truy xuất dữ liệu mà nó định nghĩa bằng câu lệnh SELECT để từ đó đưa ra các tùy chọn.
- Sau khi con trỏ được khai báo nó đã tồn tại và ta có thể truy xuất dữ liệu dựa vào các tùy chọn đã gán cho con trỏ.
- Con trỏ thường chứa dữ liệu.
- Con trỏ sau khi thực hiện xong nhiệm vụ nó phải được đóng lại.

2. TẠO CON TRỎ:

Con trỏ được tạo dựa vào câu lệnh khai báo DECLARE, đầu tiên ta khai báo tên của con trỏ sau đó là các tùy chọn thực hiện nằm trong câu lệnh SELECT và kết thúc là mệnh đề WHERE.

Ví dụ:

Ta sẽ tạo ra một con trỏ có thể truy xuất tất cả các khách hàng không có địa chỉ email như sau:

```
DECLARE CUSTCURSOR CURSOR
FOR
SELECT *
FROM CUSTOMERS
WHERE CUST_EMAIL IS NULL;
```

Nếu bạn đang sử dụng phiên bản Oracle thì cú pháp như sau:

```
DECLARE CUSTCURSOR CURSOR
IS
SELECT *
FROM CUSTOMERS
WHERE CUST_EMAIL IS NULL;
```

Trong ví dụ trên bạn đã thấy ta đã tạo ra một con trỏ có tên CUSTCURSOR, còn câu lệnh **select** định nghĩa một con trỏ có nội dung là tìm tất cả các khách hàng không có địa chỉ email.

3. SỬ DỤNG CON TRỎ:

Bạn có thể dùng câu lệnh OPEN CURSOR để mở con trỏ ra sử dụng, cú pháp:

```
OPEN CURSOR CUSTCURSOR
```

Khi con trỏ được mở thì nó sẽ thực thi các câu lệnh có trong tùy chọn của nó và dữ liệu sẽ được truy xuất ra.

Bây giờ dữ liệu của con trỏ có thể được truy xuất khi ta sử dụng câu lệnh FETCH, nó là các dòng để truy xuất dữ liệu và có thể là nơi lưu trữ.

Vi du:

```
DECLARE     @CUST_ID CHAR(10),
            @CUST_NAME CHAR(50),
            @CUST_ADDRESS CHAR(50),
            @CUST_CITY CHAR(50),
            @CUST_STATE CHAR(5),
            @CUST_ZIP CHAR(10),
            @CUST_COUNTRY CHAR(50),
            @CUST_CONTACT CHAR(50),
            @CUST_EMAIL CHAR(255)
OPEN CUSTCURSOR
FETCH NEXT FROM CUSTCURSOR
      INTO  @CUST_ID,
            @CUST_NAME,
            @CUST_ADDRESS,
            @CUST_CITY,
            @CUST_STATE,
            @CUST_ZIP,
            @CUST_COUNTRY,
            @CUST_CONTACT,
            @CUST_EMAIL
WHILE @@FETCH_STATUS=0
BEGIN
FETCH NEXT FROM CUSTCURSOR
      INTO  @CUST_ID,
            @CUST_NAME,
            @CUST_ADDRESS,
            @CUST_CITY,
            @CUST_STATE,
            @CUST_ZIP,
```

```
                @CUST_COUNTRY,
                @CUST_CONTACT,
                @CUST_EMAIL
    END
    CLOSE CUSTCURSOR;
```

Trong ví dụ trên có rất nhiều biến được khai báo tương ứng cho mỗi cột truy xuất và câu lệnh FETCH truy xuất một dòng sau đó lưu các giá trị vào các biến này. Trong khi câu lệnh lặp While được thực hiện để lặp lại các dòng, điều kiện @@FETCH_STATUS=0 để kết thúc vòng lặp khi mà không tìm thêm vòng nữa.

4. ĐÓNG CON TRỎ:

Sau khi sử dụng bạn phải đóng con trỏ lại theo cú pháp sau:

```
CLOSE CUSTSOR
DEALLOCATE CURSOR CUSTCURSOR
```

Nếu bạn dùng Oracle thì cú pháp:

```
CLOSE CUSTCURSOR
```

BÀI 22

TÌM HIỂU MỘT SỐ ĐẶC TRƯNG CỦA SQL

Trong bài học này bạn sẽ được hiểu thêm về nhiều đặc trưng tối ưu của SQL như: constraints, indexes và triggers.

1. TÌM HIỂU VỀ RÀNG BUỘC (Contraints):

Thực ra contraints là tập hợp các nguyên tắc để quản lí dữ liệu được chèn tự động hay thủ công.

a) Khóa chính (primary keys):

Bất kì cột nào trong một bảng đều được nhận dạng (thiết lập) bởi một khóa chính, với các nguyên tắc sau:

- Trong một bảng không thể có 2 dòng có khóa chính giống nhau.
- Mỗi dòng phải có một khóa chính.
- Cột chứa những giá trị khóa chính sẽ không bị sửa đổi hay cập nhật và phải mang giá trị NOT NULL.
- Giá trị khóa chính có thể không được dùng lại, nếu một dòng bị xóa ra khỏi bảng thì khóa chính của nó sẽ không được gán cho bất cứ dòng mới nào cả.

Sau đây là một cách để định nghĩa khóa chính:

```
CREATE TABLE VENDORS
(
VEND_ID          CHAR(10)    NOT NULL  PRIMARY KEY,
VEND_NAME        CHAR(50)    NOT NULL,
VEND_ADDRESS     CHAR(50),
VEND_CITY        CHAR(50),
VEND_STATE       CHAR(5),
VEND_ZIP         CHAR(10),
VEND_COUNTRY     CHAR(50)
);
```

Trong ví dụ trên ta thấy bảng vendors có khóa chính là cột vend_id.

Hoặc bạn có thể định nghĩa khóa chính cho cột vend_id như sau:

```
ALTER TABLE VENDORS
ADD CONSTRAINT PRIMARY KEY (VEND_ID)
```

b) Khóa phụ (Foreign):

Khóa phụ là một cột trong bảng mà giá trị của nó có trong một bảng khác.

Ví dụ:

```
CREATE TABLE ORDERS
(
  ORDER_NUM   INTEGER     NOT NULL PRIMARY KEY,
  ORDER_DATE  DATETIME    NOT NULL,
  CUST_ID     CHAR(10)    NOT NULL REFERENCES
  CUSTOMERS(CUST_ID)
);
```

Trong ví dụ trên ta đã định nghĩa một khóa phụ bằng từ khóa references để cho biết rằng bất cứ giá trị nào của cust_id của bảng orders phải được đặt trong cust_id của bảng customers.

Hoặc ta có thể thực hiện gán như sau cho cùng kết quả:

```
ALTER TABLE ORDERS
CONSTRAINT FOREIGN KEY (CUST_ID) REFERENCES
CUSTOMERS(CUST_ID)
```

c) Sự ràng buộc duy nhất (unique constraints):

Thực chất nó được sử dụng để bảo đảm rằng tất cả dữ liệu trong cột (hay một tập hợp của cột) là duy nhất. Nó tương đối giống khóa chính song sau đây là một số khác biệt:

- Mỗi bảng có thể bao gồm nhiều unique constraints song chỉ có duy nhất một khóa chính.
- Cột unique constraint có thể chứa giá trị NULL, có thể sửa đổi hay cập nhật và có thể được dùng lại.
- Unique constraint không thể được sử dụng để định nghĩa khóa phụ.

Ví dụ trong bảng nhân viên mỗi người có một số security ID cho riêng mình nhưng bạn không muốn sử dụng nó làm khóa chính bởi vì nó có quá nhiều. Vì vậy bạn phải dùng constraint bằng cách này bạn có thể định nghĩa cho nó một unique constraint trong cột security IDs.

Cú pháp sử dụng rất đơn giản bạn sẽ dùng từ khóa UNIQUE để định nghĩa hay một CONSTRAINT cũng được.

d) Kiểm tra constraints:

Kiểm tra sự ràng buộc là cách được sử dụng để chắc rằng tất cả dữ liệu trong cột (hay tập hợp cột) thì phù hợp với tiêu chuẩn bạn đặt ra:

- Kiểm tra giá trị nhỏ nhất hay lớn nhất, ví dụ kiểm tra để

không cho một hóa đơn không có hàng hóa kể cả đó là số 0.

- Ghi rõ sự sắp xếp. Ví dụ bạn phải chắc rằng ngày chuyển hàng phải trễ hơn hay chí ít cũng là hôm nay.
- Cho phép các giá trị cụ thể.

Ví dụ bây giờ bạn hãy kiểm tra xem trong bảng Orderitems chắc rằng tất cả các hàng hóa có quantity (số lượng) nhiều hơn 0 đơn vị:

```
CREATE TABLE ORDERITEMS
(
ORDER_NUM   INTEGER    NOT NULL,
ORDER_ITEM  INTEGER    NOT NULL,
PROD_ID     CHAR(10)   NOT NULL,
QUANTITY    INTEGER    NOTNULL  CHECK(QUANTITY>0),
ITEM_PRICE  DECIMAL(8,2)NOT NULL
);
```

Với sự ràng buộc được đặt thì bất cứ dòng nào khi được chèn vào sẽ bị kiểm tra xem có thỏa mãn không, tức là ràng buộc quantity lớn hơn 0. Để kiểm tra xem một cột bạn đặt tên gender có các từ M hay F không chẳng hạn thì bạn làm như sau:

```
CONSTRAINT CHECK (GENDER LIKE '[MF]' )
```

2. TÌM HIỂU VỀ INDEXES:

Thực ra indexes được thiết kế dùng để sắp xếp dữ liệu để từ đó hỗ trợ hàm Sort thực hiện một cách nhanh nhất.

Ta có thể tạo một index(chỉ mục) với câu lệnh sau:

```
CREATE INDEX <tên chỉ mục>
```

Sau đây là một ví dụ đơn giản giúp bạn tạo một chỉ mục cho bảng sản phẩm của cột tên sản phẩm:

```
CREATE INDEX PROD_NAME_IND
ON PRODUCTS(PROD_NAME);
```

Mỗi index chỉ có một tên duy nhất. Trong ví dụ trên tên của

chỉ mục là prod_name_ind được định nghĩa sau từ khóa CREATE INDEX.ON được sử dụng để chỉ rõ bảng được tạo chỉ mục và các cột sẽ tạo index.

3. TÌM HIỂU VỀ TRIGGERS:

Triggers là một kiểu stored procedures dạng đặc biệt, nó sẽ thực thi một cách tự động khi dữ liệu trên đường truyền bị hư. Nó có thể kết hợp với các toán tử INSERT, UPDATE và DELETE trong bảng.

Sau đây là một ví dụ về Triggers, nó mô tả cách chuyển đổi cột cust_state có trong bảng customers để nở hoa tất cả đối tượng khi được chèn (insert) hay cập nhật vào bảng (update):

```
CREATE TRIGGER CUSTOMER_STATE
ON CUSTOMERS
FOR INSERT,UPDATE
AS
UPDATE CUSTOMERS
SET CUST_STATE=UPPER(CUST_STATE)
WHERE CUSTOMERS.CUST_ID=INSERTED.CUST_ID;
```

Nếu bạn dùng phiên bản Oracle thì cú pháp như sau:

```
CREATE TRIGGER CUSTOMER_STATE
AFTER INSERT OR UPDATE
FOR EACH ROW
BEGIN
UPDATE CUSTOMERS
SET CUST_STATE=UPPER(CUST_STATE)
WHERE CUSTOMERS.CUST_ID=:OLD.CUST_ID
END;
```

PHỤ LỤC A

CÁC VÍ DỤ SỬ DỤNG TRONG SÁCH:

1. Bảng Vendors (Đại lí):

a) Tạo bảng Vendors (Đại lí):

```
CREATE TABLE VENDORS
(
    VEND_ID         CHAR(10)    NOT NULL,
    VEND_NAME       CHAR(50)    NOT NULL,
    VEND_ADDRESS    CHAR(50),
    VEND_CITY       CHAR(50),
    VEND_STATE      CHAR(5),
    VEND_ZIP        CHAR(10),
    VEND_COUNTRY    CHAR(50)
);
```

b) Nhập liệu vào bảng Vendors:

```
INSERT INTO VENDORS
(
    VEND_ID,
    VEND_NAME,
    VEND_ADDRESS,
    VEND_CITY,
    VEND_STATE,
    VEND_ZIP,
    VEND_COUNTRY
)
```

```
        VALUES
(
        'BRS01',
        'Bears R us',
        '123 Main Street',
        'Bear Town',
        'MI',
        '44444',
        'USA'
);
        INSERT INTO VENDORS
(
        VEND_ID,
        VEND_NAME,
        VEND_ADDRESS,
        VEND_CITY,
        VEND_STATE,
        VEND_ZIP,
        VEND_COUNTRY
)
        VALUES
(
        'BRS02',
        'Bear Emporium'      ,
        '500 Park Street'
        'Any town'     ,
        'OH'    ,
        '44333' ,
        'USA'
);
```

```
INSERT INTO VENDORS
(
VEND_ID,
VEND_NAME,
VEND_ADDRESS,
VEND_CITY,
VEND_STATE,
VEND_ZIP,
VEND_COUNTRY
)
VALUES
(
'DLL01'
'Doll House Inc.
'555 High Street'
'Dollsville'
'CA'
'99999' ,
'USA'
);
INSERT INTO VENDORS
(
VEND_ID,
VEND_NAME,
VEND_ADDRESS,
VEND_CITY,
VEND_STATE,
VEND_ZIP,
VEND_COUNTRY
)
```

```
        VALUES
(
        'FRB01',
        'Furball Inc.'    ,
        '1000 5th Avenue'       ,
        'New York'     ,
        'NY'    ,
        '11111' ,
        'USA'
);
        INSERT INTO VENDORS
(
        VEND_ID,
        VEND_NAME,
        VEND_ADDRESS,
        VEND_CITY,
        VEND_STATE,
        VEND_ZIP,
        VEND_COUNTRY
)
        VALUES
(
        'FNG01'         ,
        'Fun and Games',
        '42 Galaxy Road'        ,
        'London'        ,
        ''      ,
        'N16 6PS'       ,
        'England'
);
```

```
INSERT INTO VENDORS
(
    VEND_ID,
    VEND_NAME,
    VEND_ADDRESS,
    VEND_CITY,
    VEND_STATE,
    VEND_ZIP,
    VEND_COUNTRY
)
    VALUES
(
    'JTS01' ,
    'Jouets et ours',
    '1 Rue Amusement'   ,
    'Paris'  ,
    ''       ,
    '45678' ,
    'France'
);
```

Toàn bộ bảng được mô tả như sau:

	VEN...	VEN...	...DRESS	VEND CITY	VEND STATE	VEN...	
1		Bears Rus	123 Main Street	Bear Town	MI	44444	USA
2	BRS01	Bear Emporium	500 Park Street	Any town	OH	44333	USA
3	DLL01	Doll House Inc	555 High Street	Dollsville	CA	99999	USA
4	FRB01	Furball Inc	1000 5th Avenue	New York	NY	11111	USA
5	FNG01	Fun and Games	42 Galaxy Road	London		N16 6ps	England
6	JTS01	Jouets et ours	1 Rue Amusement	Paris		45678	France

2. Bảng Products (sản phẩm):

a) Tạo bảng Products (sản phẩm):

```
CREATE TABLE PRODUCTS
(
    PROD_ID         CHAR(10)        NOT NULL,
    VEND_ID         CHAR(10)        NOT NULL,
    PROD_NAME       CHAR(255)       NOT NULL,
    PROD_PRICE      DECIMAL(8,2)    NOT NULL,
    PROD_DESC       VARCHAR(1000)
);
```

b) Nhập liệu vào bảng Products:

```
INSERT INTO PRODUCTS
(
    PROD_ID,
    VEND_ID,
    PROD_NAME,
    PROD_PRICE,
    PROD_DESC
)
    VALUES
(
    'BR01',
    'BRS01',
    '8 inch teddy bear',
    5.99,
    '8 inch teddy bear,comes with cap and jacket'
);
```

```
        INSERT INTO PRODUCTS
(
        PROD_ID,
        VEND_ID,
        PROD_NAME,
        PROD_PRICE,
        PROD_DESC
)
        VALUES
(
        'BR02',
        'BRS01',
        '12 inch teddy bear',
        8.99,
        '12 inch teddy bear,comes with cap and jacket'
);
        INSERT INTO PRODUCTS
(
        PROD_ID,
        VEND_ID,
        PROD_NAME,
        PROD_PRICE,
        PROD_DESC
)
        VALUES
(
        'BR01',
        'BRS01',
        '8 inch teddy bear',
        5.99,
        '8 inch teddy bear,comes with cap and jacket'
);
```

```
        INSERT INTO PRODUCTS
(
        PROD_ID,
        VEND_ID,
        PROD_NAME,
        PROD_PRICE,
        PROD_DESC
)
        VALUES
(
        'BR03',
        'BRS01',
        '18 inch teddy bear',
        11.99,
        '18 inch teddy bear,comes with cap and jacket'
);
        INSERT INTO PRODUCTS
(
        PROD_ID,
        VEND_ID,
        PROD_NAME,
        PROD_PRICE,
        PROD_DESC
)
        VALUES
(
        'BNBG01',
        'DLL01',
        'Fish bean bag toy',
        3.49,
        'Fish bean bag toy,complete with bean bag worms with
which to feed it'
);
```

```
INSERT INTO PRODUCTS
(
PROD_ID,
VEND_ID,
PROD_NAME,
PROD_PRICE,
PROD_DESC
)
VALUES
(
'BNBG02',
'DLL01',
'Bird bean bag toy',
3.49,
'Bird bean bag toy,eggs are not included'
);
INSERT INTO PRODUCTS
(
PROD_ID,
VEND_ID,
PROD_NAME,
PROD_PRICE,
PROD_DESC
)
VALUES
(
'BNBG03',
'DLL01',
'Rabbit bean bag toy',
3.49,
'Rabbit bean bag toy,comes with bean bag carrots'
);
```

```
INSERT INTO PRODUCTS
(
PROD_ID,
VEND_ID,
PROD_NAME,
PROD_PRICE,
PROD_DESC
)
VALUES
(
'RGAN01',
'DLL01',
'Raggedy Ann',
4.99,
'18 inch Raggedy Ann doll'
);
INSERT INTO PRODUCTS
(
PROD_ID,
VEND_ID,
PROD_NAME,
PROD_PRICE,
PROD_DESC
)
VALUES
(
'RYL01',
'FNG01',
'King doll',
9.49,
'12 inch king doll with royal garments and crown'
);
```

```
INSERT INTO PRODUCTS
(
    PROD_ID,
    VEND_ID,
    PROD_NAME,
    PROD_PRICE,
    PROD_DESC
)
    VALUES
(
    'RYL02',
    'FNG01',
    'Queen doll',
    9.49,
    '12 inch queen doll with royal garments and crown'
);
```

Toàn bộ bảng được mô tả như sau:

	PROD_ID	VEND_ID	PROD_NAME	PROD_PRICE	PROD_DESC
1	BR01	BRS01	8 inch tedddy bear	5.99	8 inch teddy bear, comes with cap and jacket
2	BR02	BRS01	12 inch teddy bear	8.99	12 inch teddy bear, comes with cap and jacket
3	BNBG01	DLL01	Fish bean bag toy	3.49	Fish bean bag toy, complete with bean bag worms with which to feed it
4	BNBG02	DLL01	Bird bean bag toy	3.49	Bird bean bag toy, eggs are not included
5	BNBG03	DLL01	Rabbit bean bag toy	3.49	Rabbit bean bag toy, comes with bean bag carrots
6	RGAN01	DLL01	Raggedy Ann	4.99	18 inch Raggedy Ann doll
7	RYL01	FNG01	King doll	9.49	12 inch king doll with royal garments and crown
8	RYL02	FNG01	Queen doll	9.49	12 inch king doll with royal garments and crown

3. Bảng Customers (khách hàng):

a) Tạo bảng Customers (khách hàng):

```
CREATE TABLE CUSTOMERS
(
        cust_id         char(10)        not null,
        cust_name       char(50)        not null,
        cust_address    char(50),
        cust_city       char(50),
        cust_state      char(5),
        cust_zip        char(10),
        cust_country    char(50),
        cust_contact    char(50),
        cust_email      char(255)
);
```

b) Nhập liệu vào bảng Customers:

```
INSERT INTO CUSTOMERS
(
        cust_id,
        cust_name,
        cust_address,
        cust_city,
        cust_state,
        cust_zip,
        cust_country,
        cust_contact,
        cust_email
)
```

```
VALUES
(
        '1000000001',
        'Village Toys',
        '200 Maple Lane',
        'Detroit',
        'MI',
        '44444',
        'USA',
        'John Smith',
        'sales@villagetoys.com'
);

INSERT INTO CUSTOMERS
(
        cust_id,
        cust_name,
        cust_address,
        cust_city,
        cust_state,
        cust_zip,
        cust_country,
        cust_contact,
        cust_email
)
```

```
VALUES
(
        '1000000002',
        'Kids Place',
        '333 South Lake Drive',
        'Columbus',
        'OH',
        '43333',
        'USA',
        'Michelle Green',
        ''
);

INSERT INTO CUSTOMERS
(
        cust_id,
        cust_name,
        cust_address,
        cust_city,
        cust_state,
        cust_zip,
        cust_country,
        cust_contact,
        cust_email
)
```

```
VALUES
(
        '1000000003',
        'Fun4All',
        '1 Sunny Place',
        'Muncie',
        'IN',
        '42222',
        'USA',
        'Jim Jones',
        'jjones@fun4all.com'
);

INSERT INTO CUSTOMERS
(
        cust_id,
        cust_name,
        cust_address,
        cust_city,
        cust_state,
        cust_zip,
        cust_country,
        cust_contact,
        cust_email
)
```

```
VALUES
(
        '1000000004',
        'Fun4All',
        '829 Riverside Drive',
        'Phoenix',
        'AZ',
        '88888',
        'USA',
        'Denise L. Stephens',
        'dstephens@fun4all.com'
);

INSERT INTO CUSTOMERS
(
        cust_id,
        cust_name,
        cust_address,
        cust_city,
        cust_state,
        cust_zip,
        cust_country,
        cust_contact,
        cust_email
)
```

```
VALUES
(
        '1000000005',
        'The Toy Store',
        '4545 53rd Street',
        'Chicago',
        'IL',
        '54545',
        'USA',
        'Kim Howard',
        ''
);
```

Toàn bộ bảng được mô tả như sau:

	cust_id	cust_name	cust_address	cust-city	cust_state
1	1000000001	Viilage Toys	200 Maple Lane	Detroit	MI
2	1000000002	Kids Place	333 South Lake Drive	Columbus	OH
3	1000000003	Fun4All	1 Sunny Place	Muncie	IN
4	1000000004	Fun4All	829 Riverside Drive	Phoenix	AZ
5	1000000005	The Toy Store	4545 53rd Street	Chicago	IL

cust_zip	cust_country	cust_contact	cust_email
44444	USA	John Smith	sales@villagetoys.com
43333	USA	Michelle Green	
42222	USA	Jim Jones	jjones@fun4all.com
88888	USA	Denise L. Stephens	dstephens@fun4all.com
54545	USA	Kim Howard	

4. Bảng Custnew (khách hàng mới):

a) Tạo bảng Custnew (khách hàng mới):

```
CREATE TABLE CUSTNEW
(
        cust_id         CHAR(10)    NOT NULL,
        cust_name       CHAR(50)    NOT NULL,
        cust_address    CHAR(50),
        cust_city       CHAR(50),
        cust_state      CHAR(5),
        cust_zip        CHAR(10),
        cust_country    CHAR(50),
        cust_contact    CHAR(50),
        cust_email      CHAR(255)
);
```

b) Nhập liệu vào bảng Custnew:

```
INSERT INTO CUSTNEW
(
        cust_id,
        cust_name,
        cust_address,
        cust_city,
        cust_state,
        cust_zip,
        cust_country,
        cust_contact,
        cust_email
)
```

```
VALUES
(
        '1000000007',
        'Lê Trần Diễm Quý',
        '12/5,Lý Thường Kiệt,Q.Tân Bình',
        'HỒ CHÍ MINH',
        'HCM',
        '55555',
        'VIỆT NAM',
        '',
        ''
);

INSERT INTO CUSTNEW
(
        cust_id,
        cust_name,
        cust_address,
        cust_city,
        cust_state,
        cust_zip,
        cust_country,
        cust_contact,
        cust_email
)
```

```
VALUES
(
        '1000000008',
        'Nguyễn Thị Hồng',
        '172 ĐINH TIÊN HOÀNG',
        'HỒ CHÍ MINH',
        'HCM',
        '55555',
        'VIỆT NAM',
        '',
        ''
);
INSERT INTO CUSTNEW
(
        cust_id,
        cust_name,
        cust_address,
        cust_city,
        cust_state,
        cust_zip,
        cust_country,
        cust_contact,
        cust_email
)
```

```
VALUES
(
        '1000000009',
        'Nguyễn Công Tuấn',
        '121,Lý Thường Kiệt,Q.Tân Bình',
        'HỒ CHÍ MINH',
        'HCM',
        '66666',
        'VIỆT NAM',
        '',
        ''
);
```

5. **Bảng Orders (hóa đơn):**

a) Tạo bảng Orders (hóa đơn):

```
CREATE TABLE ORDERS
(
        ORDER_NUM       INTEGER     NOT NULL,
        ORDER_DATE      DATETIME    NOT NULL,
        CUST_ID         CHAR(10)    NOT NULL
);
```

b) Nhập liệu vào bảng Orders:

```
INSERT INTO ORDERS
(
        ORDER_NUM,
        ORDER_DATE,
        CUST_ID
)
```

```
VALUES
(
        20005,
        '01-MAY-2004',
        '1000000001'
);
INSERT INTO ORDERS
(
        ORDER_NUM,
        ORDER_DATE,
        CUST_ID
)
VALUES
(
        20006,
        '12-JAN-2004',
        '1000000003'
);
INSERT INTO ORDERS
(
        ORDER_NUM,
        ORDER_DATE,
        CUST_ID
)
VALUES
(
        20007,
        '30-JAN-2004',
        '1000000004'
);
```

```
INSERT INTO ORDERS
(
        ORDER_NUM,
        ORDER_DATE,
        CUST_ID
)
VALUES
(
        20008,
        '03-FEB-2004',
        '1000000005'
);
INSERT INTO ORDERS
(
        ORDER_NUM,
        ORDER_DATE,
        CUST_ID
)
VALUES
(
        20009,
        '08-FEB-2004',
        '1000000001'
);
```

Toàn bộ bảng Orders được mô tả như sau:

	ORDER_NUM	ORDER_DATE	CUST_ID
1	20006	2004_01_12 00:00:00.000	1000000003
2	20005	2004_05_01 00:00:00.000	1000000001
3	20007	2004_01_30 00:00:00.000	1000000004
4	20008	2004_02_03 00:00:00.000	1000000005
5	20009	2004_02_08 00:00:00.000	1000000001

6. Bảng OrderItems:

a) Tạo bảng OrderItems:

```
CREATE TABLE ORDERITEMS
(
ORDER_NUM    INTEGER        NOT NULL,
ORDER_ITEM   INTEGER        NOT NULL,
PROD_ID      CHAR(10)       NOT NULL,
QUANTITY     INTEGER        NOT NULL,
ITEM_PRICE   DECIMAL(8,2)   NOTNULL
);
```

b) Nhập liệu vào bảng OrderItems:

```
INSERT INTO ORDERITEMS
(
        ORDER_NUM,
        ORDER_ITEM,
        PROD_ID,
        QUANTITY,
        ITEM_PRICE
)
VALUES
(
        '20005',
        1,
        'BR01',
        100,
        5.49
);
```

```
INSERT INTO ORDERITEMS
(
        ORDER_NUM,
        ORDER_ITEM,
        PROD_ID,
        QUANTITY,
        ITEM_PRICE
)
VALUES
(
        '20005',
        2,
        'BR03',
        100,
        10.99
);
INSERT INTO ORDERITEMS
(
        ORDER_NUM,
        ORDER_ITEM,
        PROD_ID,
        QUANTITY,
        ITEM_PRICE
)
VALUES
(
        '20006',
        1,
        'BR01',
        20,
        5.99
);
```

```
INSERT INTO ORDERITEMS
(
        ORDER_NUM,
        ORDER_ITEM,
        PROD_ID,
        QUANTITY,
        ITEM_PRICE
)
VALUES
(
        '20006',
        2,
        'BR02',
        10,
        8.99
);
INSERT INTO ORDERITEMS
(
        ORDER_NUM,
        ORDER_ITEM,
        PROD_ID,
        QUANTITY,
        ITEM_PRICE
)
VALUES
(
        '20006',
        3,
        'BR03',
        10,
        11.99
);
```

```
INSERT INTO ORDERITEMS
(
        ORDER_NUM,
        ORDER_ITEM,
        PROD_ID,
        QUANTITY,
        ITEM_PRICE
)
VALUES
(
        '20007',
        1,
        'BR03',
        50,
        11.49
);
INSERT INTO ORDERITEMS
(
        ORDER_NUM,
        ORDER_ITEM,
        PROD_ID,
        QUANTITY,
        ITEM_PRICE
)
VALUES
(
        '20007',
        2,
        'BNBG01',
        100,
        2.99
);
```

```
INSERT INTO ORDERITEMS
(
        ORDER_NUM,
        ORDER_ITEM,
        PROD_ID,
        QUANTITY,
        ITEM_PRICE
)
VALUES
(
        '20007',
        3,
        'BNBG02',
        100,
        2.99
);
INSERT INTO ORDERITEMS
(
        ORDER_NUM,
        ORDER_ITEM,
        PROD_ID,
        QUANTITY,
        ITEM_PRICE
)
VALUES
(
        '20007',
        4,
        'BNBG03',
        100,
        2.99
);
```

```
INSERT INTO ORDERITEMS
(
        ORDER_NUM,
        ORDER_ITEM,
        PROD_ID,
        QUANTITY,
        ITEM_PRICE
)
VALUES
(
        '20007',
        5,
        'RGAN01',
        50,
        4.49
);
INSERT INTO ORDERITEMS
(
        ORDER_NUM,
        ORDER_ITEM,
        PROD_ID,
        QUANTITY,
        ITEM_PRICE
)
VALUES
(
        '20008',
        1,
        'RGAN01',
        5,
        4.99
);
```

```
INSERT INTO ORDERITEMS
(
        ORDER_NUM,
        ORDER_ITEM,
        PROD_ID,
        QUANTITY,
        ITEM_PRICE
)
VALUES
(
        '20008',
        2,
        'BR03',
        5,
        11.99
);
INSERT INTO ORDERITEMS
(
        ORDER_NUM,
        ORDER_ITEM,
        PROD_ID,
        QUANTITY,
        ITEM_PRICE
)
VALUES
(
        '20008',
        3,
        'BNBG01',
        10,
        3.49
);
```

```
INSERT INTO ORDERITEMS
(
        ORDER_NUM,
        ORDER_ITEM,
        PROD_ID,
        QUANTITY,
        ITEM_PRICE
)
VALUES
(
        '20008',
        4,
        'BNBG02',
        10,
        3.49
);
INSERT INTO ORDERITEMS
(
        ORDER_NUM,
        ORDER_ITEM,
        PROD_ID,
        QUANTITY,
        ITEM_PRICE
)
VALUES
(
        '20008',
        5,
        'BNBG03',
        10,
        3.49
);
```

```
INSERT INTO ORDERITEMS
(
        ORDER_NUM,
        ORDER_ITEM,
        PROD_ID,
        QUANTITY,
        ITEM_PRICE
)
VALUES
(
        '20009',
        1,
        'BNBG01',
        250,
        2.49
);
INSERT INTO ORDERITEMS
(
        ORDER_NUM,
        ORDER_ITEM,
        PROD_ID,
        QUANTITY,
        ITEM_PRICE
)
VALUES
(
        '20009',
        2,
        'BNBG02',
        250,
        2.49
```

```
INSERT INTO ORDERITEMS
(
        ORDER_NUM,
        ORDER_ITEM,
        PROD_ID,
        QUANTITY,
        ITEM_PRICE
)

VALUES
(
        '20009',
        3,
        'BNBG03',
        250,
        2.49
);
```

Toàn bộ bảng Orderitems được mô tả như sau:

	ORDER_NUM	ODER_ITEM	PROD_ID	QUANTITY	ITEM_PRICE
1	20005	1	BR01	100	5.49
2	20005	2	BR03	100	10.99
3	20006	1	BR01	20	5.99
4	20006	2	BR02	10	8.99
5	20006	3	BR03	10	11.99
6	20007	1	BR03	50	11.49
7	20007	2	BNBG01	100	2.99
8	20007	3	BNBG02	100	2.99
9	20007	4	BNBG03	100	2.99
10	20007	5	BNBG01	50	4.49
11	20008	1	BNBG01	5	4.99
12	20008	2	BR03	5	11.99
13	20008	3	BNBG01	10	3.49
14	20008	4	BNBG02	10	3.49
15	20008	5	BNBG03	10	3.49
16	20009	1	BNBG01	250	2.49
17	20009	2	BNBG02	250	2.49
18	20009	3	BNBG03	250	2.49

PHỤ LỤC B

DIỄN GIẢI CÁC BẢNG VÍ DỤ

1. BẢNG VENDORS (bảng đại lí):

Bảng vendors lưu trữ tất cả các đại lí, nơi mà có các sản phẩm bán ra. Mỗi đại lí đều có một bảng ghi (record, row) trong bảng này và cột vend_id được dùng để quan hệ để truy xuất các sản phẩm với các đại lí khác.

Cột:	Diễn giải:
Vend_id	Là mã số duy nhất đại diện cho từng đại lí.
Vend_name	Tên đại lí.
Vend_address	Địa chỉ của đại lí.
Vend_city	Thành phố nơi đặt đại lí.
Vend_state	Bang nơi đại lí đặt.
Vend_zip	Mã số vùng của đại lí.
Vend_country	Đất nước nơi đặt đại lí.

2. BẢNG PRODUCTS (bảng sản phẩm):

Bảng sản phẩm chưa tất cả danh sách (catolog) các sản phẩm. Mỗi một sản phẩm là một dòng (record, row). Trong đó từng sản phẩm có một mã số nhận diện riêng (prod_id) và được liên hệ với các đại lí bằng các vend_id.

Cột:	Diễn giải:
Prod_id	Mã số duy nhất đại diện cho từng sản phẩm.
Vend_id	Tên đại lí phân phối sản phẩm này.
Prod_name	Tên sản phẩm.
Prod_price	Giá sản phẩm.
Prod_desc	Mô tả sản phẩm.

3. BẢNG CUSTOMERS (bảng khách hàng):

Bảng khách hàng lưu trữ tất cả các thông tin về khách hàng. Mỗi khách hàng có một mã số nhận dạng duy nhất gọi là cust_id và mỗi một khách hàng sẽ chiếm một dòng (record, row).

Cột:	Diễn giải:
Cust_id	Mã số nhận diện duy nhất của khách hàng.
Cust_name	Tên khách hàng.
Cust_address	Địa chỉ khách hàng.
Cust_city	Thành phố nơi khách hàng đến.
Cust_state	Bang nơi khách hàng đến.
Cust_zip	Mã số vùng của khách hàng.
Cust_country	Quốc gia nơi khách hàng đến.
Cust_contact	Tên đối tác của khách hàng.
Cust_email	Tên hộp thư điện tử của khách hàng.

4. BẢNG ORDERS (bảng hóa đơn):

Đây là nơi lưu trữ các hóa đơn của khách hàng (song không chi tiết). Mỗi hóa đơn được nhận dạng bởi một số duy nhất gọi là order_num. Các hóa đơn thì được liên hệ với các khách hàng bằng cust_id.

Cột:	Diễn giải:
Order_num	Số nhận dạng duy nhất của hóa đơn.
Order_date	Ngày nhận hóa đơn.
Cust_id	Mã số nhận dạng duy nhất của khách hàng.

5. BẢNG ORDERITEMS (bảng chi tiết hóa đơn):

Bảng này lưu trữ các mục của hóa đơn, mỗi dòng sẽ dành cho một mục của một hóa đơn. Mỗi dòng của bảng Orders có một hay nhiều dòng trong bảng orderitems. Mỗi hóa đơn sẽ được nhận dạng duy nhất bởi một số gọi là order_num và nó cũng dùng để liên kết với bảng orders.

Cột:	Diễn giải:
Order_num	Số nhận dạng hóa đơn.
Order_item	Số khoản có trong hóa đơn.
Prod_id	Mã số sản phẩm.
Quantity	Số món hàng.
Item_price	Đơn giá mỗi món hàng.

PHỤ LỤC C

CÁC KIỂU DỮ LIỆU TRONG SQL

1. KIỂU CHUỖI (STRING):

Một số kiểu chuỗi thông dụng:

Kiểu dữ liệu:	Diễn giải:
CHAR (số kí tự)	Cố định chiều dài của chuỗi từ 1 đến 255 kí tự.
NCHAR (số kí tự)	Đây là một kiểu đặc biệt của CHAR, nó cung cấp các kí tự UNICODE.
NVARCHAR (số kí tự)	Đây là một kiểu đặc biệt của TEXT, nó cung cấp các kí tự UNICODE.
TEXT (số kí tự) ,LONG,VARCHAR hay MEMO	Cung cấp chuỗi kí tự dài, nhiều kí tự.

2. KIỂU SỐ (NUMERIC):

Một số kiểu số thông dụng:

Kiểu dữ liệu:	Diễn giải:
BIT	Đưa ra chỉ một bit là số nhị phân 0 hay 1, đồng nghĩa với việc biểu diễn có hay không (on/off).
DECIMAL	Cố định hay thả nổi giá trị ở nhiều mức.
FLOAT	Kiểu thả nổi giá trị (số thực).
INT	Kiểu số nguyên [4 bytes là (-2147483648 đến +2147483648)].

REAL	Kiểu thả nổi 4 bytes.
SMALLINT	Kiểu số nguyên 2 bytes từ -32768 đến +32768.
TINYINT	Kiểu số nguyên 1 byte từ 0 đến 255.

3. KIỂU NGÀY GIỜ (DATE VÀ TIME):

Một số kiểu ngày giờ thông dụng:

Kiểu dữ liệu:	**Diễn giải:**
DATE	Cho ra giá trị ngày.
DATETIME	Cho ra giá trị giờ.
Hay TIMESTAMP	
SMALLDATETIME	Cho ra thời gian có giờ và phút.

4. KIỂU NHỊ PHÂN (BINARY):

Một số kiểu nhị phân thông dụng:

Kiểu dữ liệu:	**Diễn giải:**
BINARY	Cố định chiều dài dữ liệu nhị phân (lớn nhất 255 bytes đến 8000 bytes).
LONG RAW	Dữ liệu nhị phân có độ dài sẵn lên tới 2GB.
RAW	Cố định dữ liệu nhị phân có độ dài lên tới 255.
VARBINARY	Dữ liệu nhị phân có độ dài sẵn từ 255 bytes tới 8000 bytes.

PHỤ LỤC D

MỘT SỐ TRÌNH ỨNG DỤNG CỦA SQL

1. SỬ DỤNG JRUN 3.X:

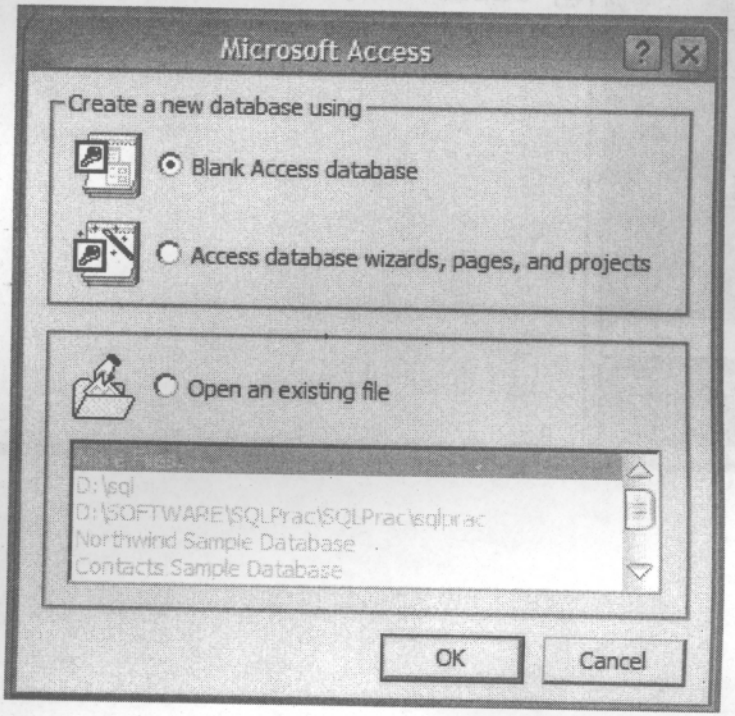
Jrun là một ứng dụng J2EE của server. Đây là một ngôn ngữ khá mạnh để viết trong Java và JSP. Nó có một thư viện có thể được sử dụng. Để chạy SQL bạn phải tạo một trang có thể chạy được bằng cách gọi nó từ trình duyệt Web browser. Các bước thực hiện như sau:

- Jrun sử dụng JDBC để tác động đến cơ sở dữ liệu, do vậy nguồn JDBC phải tồn tại trước khi thực hiện.
- Tạo một trang JSP sử dụng Jrun Studio, Homesite hay bất cứ trình soạn thảo nào đó.
- Bao gồm thư viện Jrun được sử dụng trực tiếp<%@ taglib %>.
- Sử dụng Jrun<sql> và cờ </sql> để tạo ra khối truy vấn. Đặt tên nó sử dụng id attribute và định nghĩa nguồn dữ liệu trong datasrc attribute.
- Đánh câu lệnh SQL giữa cờ <sql> và cờ </sql>.
- Sử dụng cờ Jrun<foreach> để lặp và hiển thị kết quả truy vấn.
- Lưu trang vừa tạo ra trong bất cứ thư mục nào dưới dạng tập tin chạy trong thư mục gốc của Web server.
- Chạy trang vừa tạo lên từ trình duyệt Web browser.

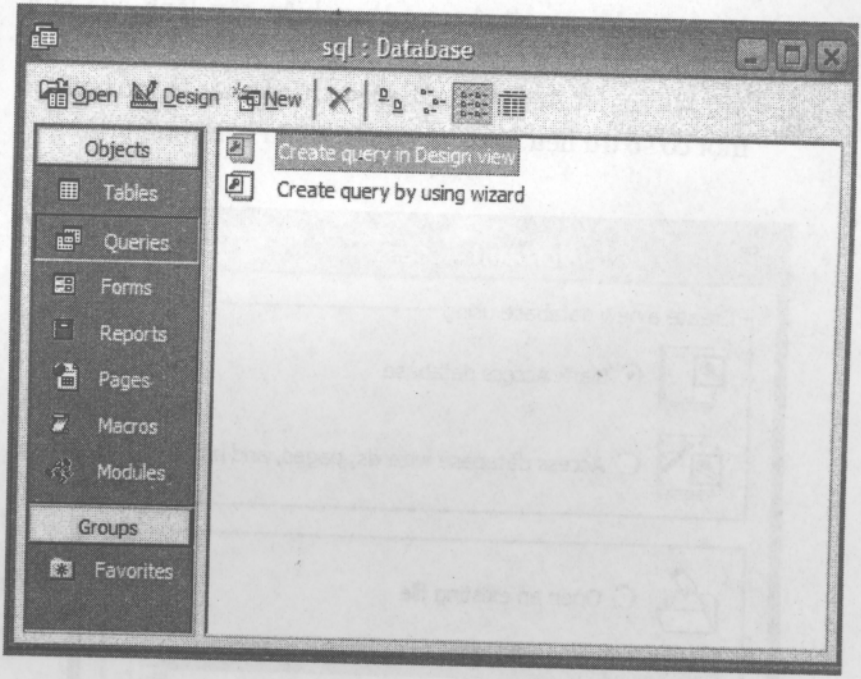
2. SỬ DỤNG MICROSOFT ACCESS:

Đây là một trình quản trị và duyệt các cơ sở dữ liệu khá phổ biến. Để sử dụng Microsoft Access thực hiện câu lệnh của SQL bạn cần làm theo các bước sau:

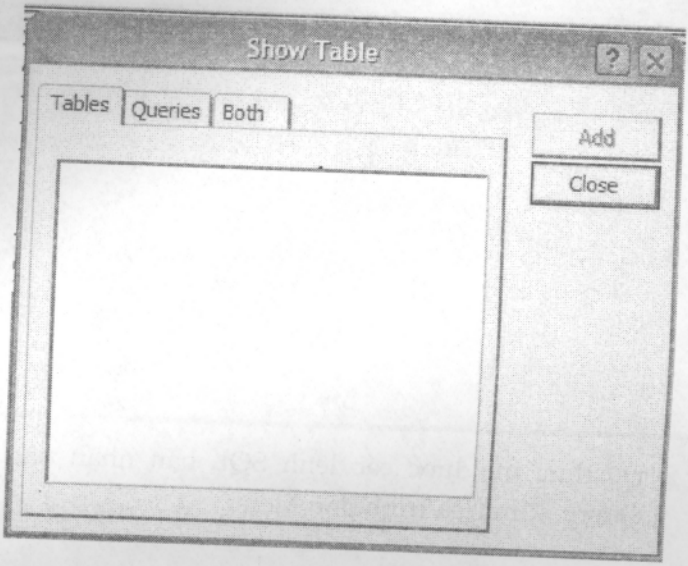
- Mở Microsoft Access lên, bạn sẽ được nhắc là mở hay tạo một cơ sở dữ liệu. Bạn hãy mở dữ liệu bạn muốn dùng.



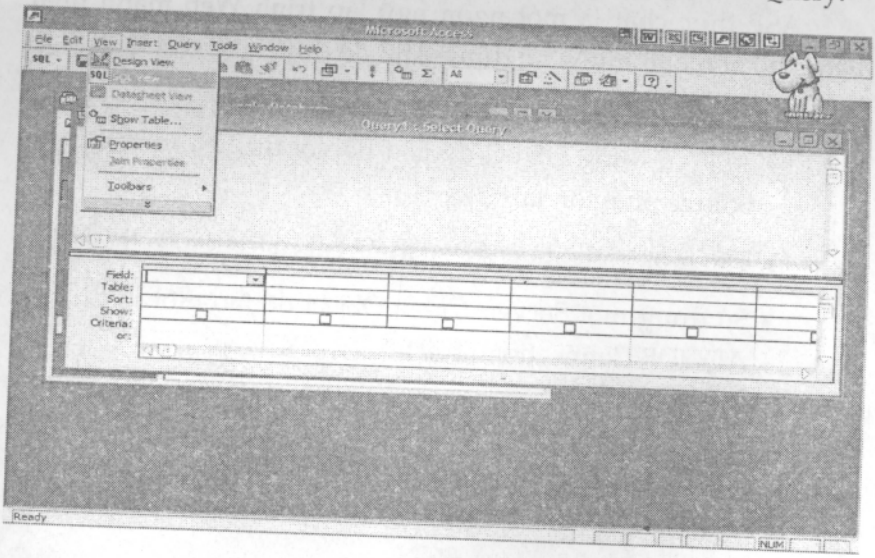
- Chọn mục Queries trong cửa sổ Database. Sau đó kích vào nút create query in Design View.



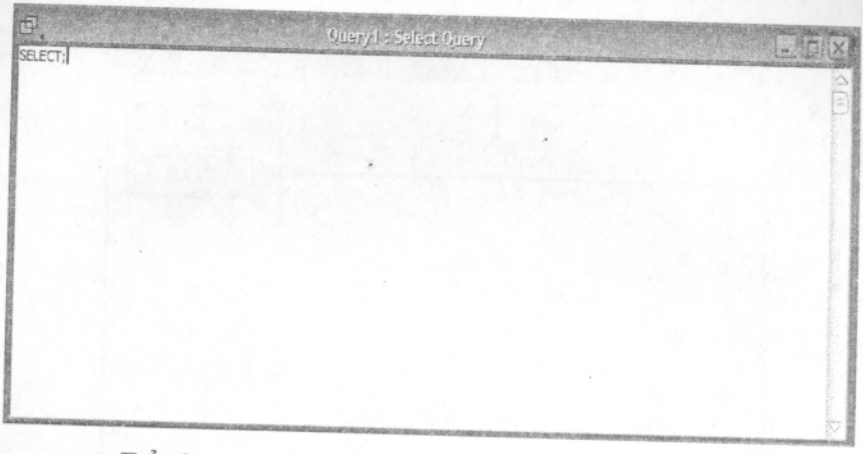
- Lúc này bạn sẽ được lưu ý là hộp thoại Show Table, bạn hãy đóng nó lại.



- Từ View Menu, chọn SQL View để hiển thị cửa sổ Query.



- Bây giờ bạn đánh câu truy vấn của bạn vào cửa sổ Query.



- Để thực thi được các lệnh SQL bạn nhấn vào menu Query, Run trên trình đơn Menu.

3. SỬ DỤNG MICROSOFT ASP:

ASP thực chất là một ngôn ngữ lập trình Web mạnh nhất hiện nay, nếu bạn muốn sử dụng ASP để duyệt các câu lệnh SQL thì làm các bước sau:

- ASP sử dụng ODBC để liên hệ với dữ liệu vì vậy Data Source phải tồn tại.
- Tạo một trang mới ASP sử dụng bất cứ trình soạn thảo nào.
- Sử dụng một Server. CreateObject để tạo một ví dụ của ADODB.Connection object.
- Hãy mở ODBC Data Source.
- Đưa câu lệnh SQL của bạn vào và thực thi.

- Để hiển thị kết quả bạn phải lặp để truy xuất dữ liệu sử dụng một vòng lặp <%Do While NOT EOF>.
- Lưu trang đó vào bất cứ thư mục gốc của Web server.
- Chạy trang đó lên bằng trình duyệt web browser.

4. SỬ DỤNG MICROSOFT SQL SERVER 2000:

Đây là phần mềm chuyên dụng để chạy SQL server, bạn chỉ cần chạy Query Analyzer có trong thư mục khi cài đặt vào là có thể thực thi được các câu lệnh SQL.

Ngoài ra bạn hãy sử dụng chương trình phần mềm chuyên dụng SQL server 6.x, SQL server 7.x,... để cài đặt vào máy vi tính và sử dụng.

Lưu ý:

SQL server có hai phiên bản, phiên bản thứ nhất dành cho hệ máy chủ (máy cài hệ điều hành server như Winnt4.0 server, Windows server 2000, Windows server 2003,...) gọi là SQL server còn phiên bản thứ hai là dành cho hệ máy trạm (client, máy chạy HĐH workstation như windows 9.x, windows 2k, xp) gọi là SQL server personal.

5. SỬ DỤNG MICROSOFT VISUAL BASIC:

Đây là một phần mềm thiết kế và xây dựng cơ sở dữ liệu rất mạnh, trong đó có hỗ trợ trình duyệt SQL. Các bước thực hiện như sau:

- VB sử dụng ODBC (và DAO) cho cơ sở dữ liệu chính vì vậy các ODBC Data Source phải tồn tại.
- Mở VB lên.
- Chọn Visual Database Manager từ trình menu.
- Từ menu File chọn Open Database và sau đó là ODBC.

- Chọn ODBC Data Source để connect tới. Bạn sẽ bị nhắc Login và Password.
- Sau khi đã connect tới ODBC Data Source bạn có thể đánh câu lệnh SQL vào cửa sổ hiển thị. Bạn muốn thực thi nó nhấn nút Execute để [illegible]

6. SỬ DỤNG ORACLE 9I:

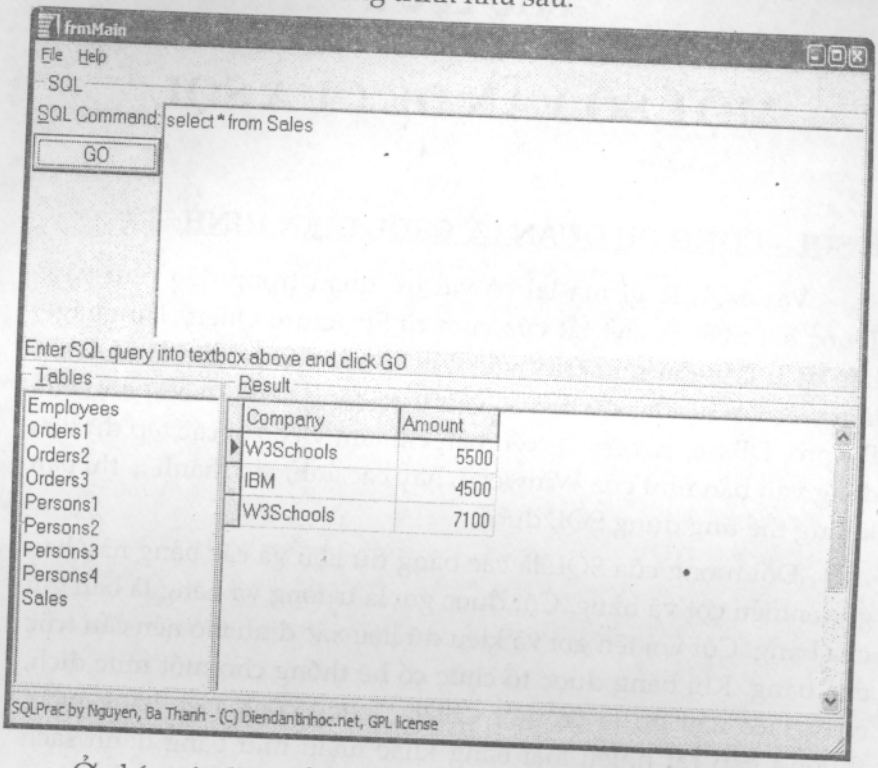
Thực ra Oracle là một trình tạo cơ sở dữ liệu cực mạnh song nó vẫn hỗ trợ truy vấn bằng các câu lệnh SQL, các bước thực hiện như sau:

- Mở Oracle lên và chọn SQL Worksheet.
- Bạn phải nhập Login và Password để vào.
- Trong Worksheet được chia làm hai phần, phần dưới dùng để nhập câu lệnh SQL vào và phần trên dùng để xuất kết quả ra. Bạn thực thi câu lệnh SQL bằng cách nhấn vào nút Execute

7. SỬ DỤNG SQLPRAC ĐỂ THỰC HÀNH CÁC CÂU LỆNH SQL:

SQLPrac là một chương trình giúp bạn thực hành các câu lệnh SQL trong bài học. SQLPrac cho phép bạn gõ vào câu lệnh SQL, thực thi câu lệnh này và xem kết quả trả về.

Giao diện của chương trình như sau:



Ở phía trên là ô để bạn gõ vào câu lệnh SQL (trong hình trên thì câu lệnh được gõ vào là select * from Sales;). Sau khi gõ lệnh, bạn click vào nút GO ở bên trái để bắt đầu thực thi câu lệnh SQL.

Kết quả của câu lệnh sẽ được hiển thị ở khung bên phải ở phía dưới. Đối với các câu lệnh không trả về kết quả (ví dụ như INSERT, DELETE,...) thì số lượng các ROW bị tác dụng sẽ được thông báo ở dòng thông báo (hiện tại dòng thông báo đang mang dòng chữ "Enter SQL query into textbox above and click GO").

Danh sách bên trái ở phía dưới liệt kê các table đang có trong database.

Trong cuốn sách này tác giả sử dụng phần mềm thông dụng và mạnh nhất hiện nay là Microsoft SQL Server 2000.

PHỤ LỤC E

MỘT SỐ VẤN ĐỀ CỦA SQL

1. SQL - CÔNG CỤ QUẢN LÝ CSDL ĐIỂN HÌNH:

Vậy SQL là gì mà lại có vai trò quan trọng đến như vậy? Trước hết SQL là viết tắt của cụm từ Structure Query Language, tạm dịch là ngôn ngữ truy vấn có cấu trúc. Có nghĩa là SQL chỉ làm việc với những dữ liệu có cấu trúc dạng bảng (table) như của Foxpro, DBase, Access ... Nếu bạn chỉ làm việc với các tệp dữ liệu dạng văn bản như của Winword, hay các ảnh, âm thanh ... thì bạn không thể ứng dụng SQL được.

Đối tượng của SQL là các bảng dữ liệu và các bảng này bao gồm nhiều cột và hàng. Cột được gọi là trường và hàng là bản ghi của bảng. Cột với tên gọi và kiểu dữ liệu xác định tạo nên cấu trúc của bảng. Khi bảng được tổ chức có hệ thống cho một mục đích, công việc nào đó ta có một CSDL. Trong công việc hàng ngày thường gặp rất nhiều loại bảng khác nhau như bảng danh sách nhân viên, danh sách phòng ban, bảng lương, bảng quyết toán... Khi bảng này được lưu thành CSDL thì có thể dùng SQL để phân tích, xử lý thông tin trong đó. Kết quả thực hiện của SQL thường là một bảng mới. Ví dụ ta yêu cầu SQL lấy ra từ bảng danh sách nhân viên những nhân viên có nhà ở Hà Nội thì SQL sẽ trả về một bảng mới chỉ chứa các nhân viên có nhà ở Hà Nội. SQL còn được dùng để cập nhật thông tin cho các CSDL đã có. Ví dụ như cần phải xóa tất cả các phiếu xuất kho hai năm trước đây chẳng hạn. Bạn có thể mở bảng dữ liệu phiếu xuất kho ra và tìm kiếm rồi xóa từng phiếu một cách thủ công, vừa mất thì giờ vừa có khả năng nhầm lẫn hoặc bỏ sót. Khi đó bạn nên dùng SQL với thời gian

thực hiện chỉ tính bằng giây và đảm bảo không có sự nhầm lẫn hay bỏ sót nào cả. Trong những trường hợp như vậy kết quả câu lệnh SQL không phải là bảng mới mà chính là bảng đã được bổ sung cập nhật dữ liệu theo yêu cầu. SQL luôn luôn thao tác trên các bảng. SQL dùng để phân tích, tổng hợp số liệu từ các bảng đã có sẵn, tạo nên các dạng bảng mới, sửa đổi cấu trúc, dữ liệu của các bảng đã có. Sau khi thực hiện một hay một chuỗi các lệnh SQL là có thể có đủ số liệu để tạo ra báo cáo theo yêu cầu công việc. Vì mục tiêu của các chương trình quản lý CSDL là quản lý dữ liệu được lưu trữ trong các bảng nên SQL được dùng rất thường xuyên trong mọi công việc.

Một trong những lý do khiến SQL ngày càng phổ biến hơn là SQL rất dễ sử dụng. Bạn không nhất thiết phải là chuyên gia tin học, cũng không cần phải biết lập trình mà vẫn có thể ứng dụng SQL để xử lý công việc của mình. Thực hiện được điều đó là nhờ có các công cụ trực quan giúp tạo ra các câu lệnh SQL một cách dễ dàng, hiệu quả, đó là các Query (câu hỏi truy vấn) trong các phần mềm có sử dụng SQL. Mỗi Query là một câu lệnh SQL được xây dựng hoàn chỉnh và ghi lại để có thể mang ra sử dụng bất cứ lúc nào. Sản phẩm của một Query là một bảng mới thì bảng này lại có thể được sử dụng trong các Query khác. Cứ như vậy ta có thể có một chuỗi các Query trước khi đi đến kết quả cuối cùng. Các Query này được thiết kế đặc biệt để thể hiện các câu hỏi của bạn về dữ liệu đã lưu trữ và khi chạy nó sẽ trả về kết quả theo yêu cầu của bạn. Sử dụng Query rất dễ dàng và ít có khả năng mắc lỗi, thậm chí bạn chỉ cần dùng chuột để tạo ra một câu lệnh SQL hoàn chỉnh. Yêu cầu duy nhất để sử dụng Query là bạn phải nắm vững cấu trúc CSDL của mình.

Khi thiết kế Query bạn chỉ cần dùng chuột chọn các bảng dữ liệu cần thiết đưa vào Query rồi chọn các trường sẽ đưa ra. Sau đó ghi Query lại với một tên nào đó và thế là bạn có thể sử dụng nó ngay lập tức hoặc dùng nó để xây dựng các Query khác.

Chúng ta sẽ cùng xây dựng một lệnh SQL bằng Query để có thể hiểu rõ hơn công dụng của nó. Giả sử bạn đã có hai bảng dữ liệu là Nhân viên và Hóa đơn trong MS Access. Bảng Nhân viên chứa danh sách các nhân viên trong công ty còn bảng Hóa đơn lưu lại tất cả các hóa đơn đã sử dụng khi bán hàng. Trường Mã nhân viên dùng để kết nối giữa hai bảng. Yêu cầu là đưa ra báo cáo số tiền bán hàng của từng nhân viên. Ta sẽ xây dựng một Query để thực hiện yêu cầu trên.

Tạo Query gồm bảng Hóa đơn và bảng Nhân viên kết hợp lại để có được tổng số tiền bán hàng theo từng tên nhân viên. Sau khi chọn New Query, hộp thoại chọn bảng sẽ hiện ra và ta chọn hai bảng cần thiết bằng cách nhấn đúp lên tên từng bảng hoặc nhấn phím Add sau khi đã chọn bảng.

Chọn bảng xong ta chọn các trường cần thiết đưa ra kết quả bằng cách nhấn đúp chuột lên tên trường đó. Các trường được chọn sẽ xuất hiện trong bảng lưới phía dưới. Cụ thể ở đây ta nhấn đúp lên trường Tên nhân viên và Thành tiền. Do cần có tổng số tiền bán hàng của từng nhân viên nên ta phải đánh dấu Totals từ menu View rồi chọn Group by cho Tên nhân viên và Sum cho Thành tiền và kết quả như trên hình.

Như vậy là bạn đã hoàn thành việc thiết kế một Query và có thể xem câu lệnh SQL đã được tự động tạo ra bằng cách chọn SQL trong menu View rồi đặt tên Query với một tên tùy ý.

Sau đó, khi muốn xem kết quả câu lệnh SQL đã có, bạn chỉ cần cho chạy Query vừa tạo ra bằng cách nhấn đúp lên nó hoặc bấm lên nút Open. Kết quả hiển thị ra màn hình là tổng hợp những dữ liệu mới nhất trong hai bảng dữ liệu nguồn tại thời điểm chạy Query.

Sau khi sử dụng thành thạo SQL bạn đã có trong tay một công cụ mạnh để xây dựng chương trình. Khi phải học sử dụng một ngôn ngữ mới nào đó thì bạn không phải học lại từ đầu cách

sử dụng SQL trong ngôn ngữ đó. Phần cốt lõi của SQL là giống nhau trong mọi ngôn ngữ. Bạn chỉ cần học cách áp dụng SQL trong từng trường hợp cụ thể của ngôn ngữ đó cùng những điểm cải tiến bổ sung mà thôi.

SQL có khả năng đáp ứng hầu hết các yêu cầu đối với việc cập nhật, phân tích dữ liệu từ các bảng. Sau đây chúng ta sẽ điểm qua một số công dụng chính của SQL:

- Chọn lọc một số cột nhất định trong bảng dữ liệu: Thường ta không sử dụng tất cả các thông tin của bảng cùng một lúc. Có thể dùng SQL để tách ra chỉ những cột cần thiết mà thôi.
- Lọc các bản ghi theo những tiêu chuẩn khác nhau: như tách riêng các hóa đơn của một khách hàng nào đó, hay in danh sách nhân viên chỉ của một vài phòng ban...
- Sắp xếp các bản ghi theo những tiêu chuẩn khác nhau: Mỗi loại báo cáo thường có yêu cầu sắp xếp các bản ghi theo những cột khác nhau để tiện cho việc theo dõi. Có báo cáo thống kê sắp xếp theo khách hàng, báo cáo khác lại sắp xếp theo mặt hàng được bán, mặc dù tất cả thông tin nằm trong cùng một bảng Bán hàng. Ta có thể thực hiện sắp xếp theo một hoặc nhiều cột khác nhau bằng SQL.

Cập nhật, xóa các bản ghi trên toàn bảng theo những điều kiện khác nhau: ví dụ như khi cần xóa toàn bộ các hóa đơn đã phát hành cách đây 5 năm...

- Kết hợp hai hay nhiều bảng theo chiều ngang: Trong CSDL, mỗi bảng lưu trữ thông tin về một đối tượng và các bảng liên hệ với nhau qua các trường khóa. Dùng SQL để thực hiện việc kết hợp các bảng này với nhau thông qua các trường khóa như ở ví dụ trên để có được bảng kết quả theo yêu cầu.
- Nối hai hay nhiều bảng theo chiều dọc: khi dữ liệu rất lớn hoặc phân tán ở nhiều nơi ta có thể phải quản lý

nhiều bảng theo cùng một mẫu, như mỗi bảng cho một quý, tháng hoặc một công ty. Khi cần tổng hợp dữ liệu của cả năm hoặc của cả tổng công ty ta có thể dùng SQL để nối các bảng lại với nhau.

- Tạo bảng mới, thay đổi cấu trúc bảng đã có: phục vụ cho việc lập trình.
- Thực hiện các phép tính toán thống kê theo từng nhóm: tổng, trung bình, max, min... Đây là chức năng thường xuyên được sử dụng để tổng hợp thông tin trước khi in báo cáo như tính tổng số lượng của từng mặt hàng, số hàng đã bán cho từng khách hàng...
- Kết nối với dữ liệu trên máy chủ (Server): Khi kho dữ liệu được tập trung trên máy chủ trong MS SQL Server hay Oracle ... ta phải dùng lệnh SQL để trực tiếp thâm nhập vào cơ sở dữ liệu.
- Kết hợp các trang Web với CSDL bằng lệnh SQL. Có thể phân ra hai loại câu lệnh SQL, một loại tổng hợp dữ liệu sang bảng mới (bảng mới có thể chỉ tồn tại trong bộ nhớ hoặc được ghi ra đĩa) và một loại chỉ cập nhật dữ liệu, cấu trúc của các bảng đã có. Bảng mới được tạo ra từ câu lệnh SQL có thể được sử dụng ở mọi nơi trong chương trình có yêu cầu dữ liệu dạng bảng, đặc biệt dùng cho báo cáo (report), danh sách (list), đối tượng lưới (grid) ...
- Tất cả các chức năng của SQL đều có thể được thực hiện bằng các công cụ khác của các phần mềm có sử dụng SQL. Tuy nhiên dùng SQL có nhiều ưu điểm lớn so với các cách khác như là:

  + Câu lệnh đơn giản, dễ nhớ, dễ sử dụng và rất ngắn gọn.
  + Tốc độ rất cao do được tối ưu hóa với những công nghệ mới.
  + Khả năng thực hiện những yêu cầu phức tạp của công việc...

- Nếu đã thành thạo lập trình xử lý các công việc, bạn nên tiết kiệm thời gian và sức lực bằng cách sử dụng SQL.
- Câu lệnh SQL ngắn gọn và đơn giản hơn thì khả năng mắc lỗi ít hơn và tốc độ vận hành nhanh hơn. Trong các ngôn ngữ như Visual Foxpro hay Access bạn được cung cấp công cụ trực quan để tạo ra Query - câu lệnh SQL được ghi lại độc lập. Sử dụng các công cụ tạo Query bạn có thể không cần phải nắm vững cú pháp câu lệnh SQL mà vẫn tạo được các Query phức tạp. Sau khi tạo xong câu lệnh, SQL sẽ được tự động sinh ra và bạn có thể sao chép dùng vào nhiều việc khác.
- Có thể bạn đang tự hỏi nếu SQL đúng là vạn năng như vậy sao không dùng một mình SQL để xử lý dữ liệu mà vẫn còn phải dùng các công cụ lập trình khác. Tuy SQL có nhiều khả năng nhưng trong một số trường hợp cụ thể vẫn phải dùng kết hợp với những công cụ lập trình khác để có được hiệu quả tối đa. Ví dụ như khi bạn cần phải làm việc với từng bản ghi chứ không phải trên phạm vi toàn bảng, khi bạn đang làm việc với một bản ghi nào đó và muốn chuyển sang làm việc với một bản ghi ở sau nó, hoặc bạn muốn có được các thông tin về cấu trúc một bảng (tên cột, kiểu dữ liệu, có khả năng cập nhật không...).
- Khả năng của SQL là rất lớn nên nó có phạm vi ứng dụng rộng trong việc quản lý CSDL bằng máy tính.

2. <u>Oracle9i: Đơn giản hóa cú pháp SQL:</u>

Oracle đã thực hiện một số thay đổi quan trọng trong Oracle8i SQL. Những thay đổi này bao gồm các lược đồ thi hành (câu lệnh) mới, hỗ trợ subquery vô hướng (scalar subquery), hỗ trợ chuẩn SQL ISO 99. Như là hệ quả, cú pháp của câu lệnh SQL cũng thay đổi theo hướng dễ sử dụng hơn khi làm việc với cơ sở dữ liệu. Dưới đây là một số thay đổi điển hình trong cú pháp của

câu lệnh SQ:

- *CROSS JOIN* - Tạo một tập tích Đề các (cặp sắp xếp có thứ tự) các hàng trong 2 bảng, giống như Oracle8i khi kết nối bảng nhưng bỏ qua mệnh đề WHERE.
- *NATURAL JOIN* - Đây là đặc điểm cú pháp rất tiện lợi trong Oralce 9i, tăng tính dễ đọc của câu lệnh SQL bằng cách loại bỏ tiêu chuẩn kết nối trong mệnh đề WHERE.
- *Mệnh đề USING* - Cho phép bạn xác định khóa kết nối theo tên.
- *Mệnh đề ON* - Cho phép xác định các cột cho các khóa kết nối trong cả 2 bảng.
- *LEFT OUTER JOIN* - Trả về tất cả các dòng trong bảng bên trái của kết nối cùng với các giá trị tương ứng trong bảng bên phải hoặc null nếu như không có dòng tương ứng ở bảng bên phải.
- *RIGHT OUTER JOIN* - Trả về tất cả các dòng trong bảng bên phải của kết nối cùng với các giá trị tương ứng trong bảng bên trái hoặc null nếu như không có dòng tương ứng ở bảng bên phải.
- *FULL OUTER JOIN* - Trả về tất cả các dòng từ hai bảng, thay bất kỳ giá trị trống nào bằng null. Không có mệnh đề tương đương trong Oracle8i.
- Phần lớn những thay đổi này nhằm mục đích cho phép các ứng dụng không phải Oracle dễ dàng tương thích với cơ sở dữ liệu Oracle hơn. Lưu ý rằng đây chỉ là những khác biệt về cú pháp, chuẩn ISO 99 không đem lại bất kỳ tính năng mới nào cho Oracle9i SQL.

CROSS JOIN

- Trong Oracle, mệnh đề CROSS JOIN tạo ra một tích Đề các các hàng trong 2 bảng, giống như khi chúng ta bỏ qua mệnh đề WHERE khi kết nối như dưới đây:

select last_name, dept_id
from emp, depts
- Trong Oracle9i, chúng ta có thể sử dụng mệnh đề CROSS JOIN để nhận được cùng một kết quả như câu lệnh trên:
select last_name, dept_id
from emp
CROSS JOIN dept;

NATURAL JOIN

- Cú pháp NATURAL JOIN rất tiện lợi bởi vì nó tự động nhận ra khóa kết nối từ tên cột trùng nhau trong cả hai bảng. Cú pháp này đơn giản hóa Oracle9i SQL nhưng yêu cầu các cột kết nối ở hai bảng phải có cùng tên. Một điều đặc biệt là cú pháp này làm việc ngay cả khi không có mối quan hệ khóa chính/khóa ngoại ở cột kết nối của hai bảng.

Oracle8i
Select book_title, sum(quantity)
From book, sales
Where book.book_id = sales.book_id
group by book_title;
Oracle9i
Select book_title, sum(quantity)
from book
natural join sales
group by book_title;

Mệnh đề USING

Sử dụng mệnh đề USING nếu có nhiều hơn một cột có cùng ên trong hai bảng nhưng bạn không muốn kết nối trên tất cả các ột này.

Oracle8i
select dept_id, city
from departments, locations

```
where departments.location_id = location.location_id;
Oracle9i
select department_name, city
from departments
JOIN locations
USING (location_id);
```

Mệnh đề ON

Mệnh đề ON được sử dụng để kết nối hai bảng khi tên của các cột trong hai bảng không trùng nhau.

```
Oracle8i
select department_name, city
from department, location
where department.location_id = location.loc_id;
Oracle9i
select department_name, city
from department d
JOIN location l
ON (d.location_id = l.id);
```

Mutitable Join

Sử dụng khi cần kết nối nhiều bảng với nhau. Chuẩn ISO SQL 99 luôn mặc định các bảng được kết nối từ trái sang phải và điều kiện kết nối hiện tại chỉ được tham chiếu tới kết nối hiện tại và các kết nối trước đó ở bên trái.

```
Oracle8i
select emp_id, city_name, dept_name
from location l, department d, emp e
where d.location_id = l.location_id
and d.department_id = e.department_id;
Oracle9i
select emp_id, city_name, dept_name
from locations l
JOIN departments d ON (d.location_id = l.location_id)
```

JOIN employees e ON (d.department_id = e.department_id);

Cú pháp OUTER JOIN mới

Chuẩn ISO 99 bỏ tất cả các dấu cộng (+) phiền hà từ kết nối mở rộng và vì vậy các kết nối mở rộng trở thành dễ hiểu hơn.

LEFT OUTER JOIN

Trong LEFT OUTER JOIN, tất cả các hàng trong bảng bên trái được trả về, ngay cả khi không có giá trị tương ứng trong cột của bảng bên phải. Trong ví dụ dưới đây, tất cả tên nhân viên được trả về, ngay cả những nhân viên chưa được gán mã phòng.

```
Oracle8i
select last_name, dept_id
from emp e, dept d
where e.department_id = d.department_id(+);
Oracle9i
select last_name, dept_id
from emp
LEFT OUTER JOIN Dept
ON e.dept_id = d.dept_id;
```

RIGHT OUTER JOIN

Trong RIGHT OUTER JOIN, tất cả các hàng trong bảng bên trái được trả về, ngay cả khi không có giá trị tương ứng trong cột của bảng bên phải. Trong ví dụ dưới đây, tất cả tên phòng được trả về, ngay cả những phòng chưa có nhân viên.

```
Oracle8i
select last_name, d.dept_id
from employees e, departments d
where e.department_id(+) = d.department_id;
Oracle9i
select last_name, d.dept_id
from employees e
RIGHT OUTER JOIN departments d
```

ON (e.department_id = d.department_id);

3. CÁC CẤU HÌNH CƠ SỞ DỮ LIỆU CLIENT/SERVER:

Nhìn chung mọi ứng dụng cơ sở dữ liệu đều bao gồm các phần:

- Thành phần xử lý ứng dụng (Application processing components).
- Thành phần phần mềm cơ sở dữ liệu (Database software componets).
- Bản thân cơ sở dữ liệu (The database itself).

Các mô hình về xử lý cơ sở dữ liệu khác nhau là bởi các trường hợp của 3 loại thành phần nói trên định vị ở đâu. Bài viết này xin giới thiệu 5 mô hình kiến trúc dựa trên cấu hình phân tán về truy nhập dữ liệu của hệ thống máy tính Client/Server:

- Mô hình cơ sở dữ liệu tập trung (Centralized database model)
- Mô hình cơ sở dữ liệu theo kiểu file - server (File - server database model)
- Mô hình xử lý từng phần cơ sở dữ liệu (Database extract processing model)
- Mô hình cơ sở dữ liệu Client/Server (Client/Server database model)
- Mô hình cơ sở dữ liệu phân tán (Distributed database model)

a) Mô hình cơ sở dữ liệu tập trung (Centralized database model):

Trong mô hình này, các thành phần xử lý ứng dụng, phần mềm cơ sở dữ liệu và bản thân cơ sở dữ liệu đều ở trên một bộ xử lý.

Ví dụ người dùng máy tính cá nhân có thể chạy các chương trình ứng dụng có sử dụng phần mềm cơ sở dữ liệu Oracle để truy nhập tới cơ sở dữ liệu nằm trên đĩa cứng của máy tính cá nhân đó. Từ khi các thành phần ứng dụng, phần mềm cơ sở dữ liệu và bản thân cơ sở dữ liệu cùng nằm trên một máy tính thì ứng dụng đã thích hợp với mô hình tập trung.

Hầu hết công việc xử lý luồng thông tin chính được thực

hiện bởi nhiều tổ chức mà vẫn phù hợp với mô hình tập trung. Ví dụ một bộ xử lý mainframe chạy phần mềm cơ sở dữ liệu IMS hoặc DB2 của IBM có thể cung cấp cho các trạm làm việc ở các vị trí phân tán sự truy nhập nhanh chóng tới cơ sở dữ liệu trung tâm. Tuy nhiên trong rất nhiều hệ thống như vậy, cả ba thành phần của ứng dụng cơ sở dữ liệu đều thực hiện trên cùng một máy mainframe do vậy cấu hình này cũng thích hợp với mô hình tập trung.

b) Mô hình cơ sở dữ liệu theo kiểu file - server (File - server database model):

Trong mô hình cơ sở dữ liệu theo kiểu file - server các thành phần ứng dụng và phần mềm cơ sở dữ liệu ở trên một hệ thống máy tính và các file vật lý tạo nên cơ sở dữ liệu nằm trên hệ thống máy tính khác. Một cấu hình như vậy thường được dùng trong môi trường cục bộ, trong đó một hoặc nhiều hệ thống máy tính đóng vai trò của server, lưu trữ các file dữ liệu cho hệ thống máy tính khác xâm nhập tới. Trong môi trường file - server, phần mềm mạng được thi hành và làm cho các phần mềm ứng dụng cũng như phần mềm cơ sở dữ liệu chạy trên hệ thống của người dùng cuối xem các file hoặc cơ sở dữ liệu trên file server thực sự như là trên máy tính của người chính họ.

Mô hình file server rất giống với mô hình tập trung. Các file cơ sở dữ liệu nằm trên máy khác với các thành phần ứng dụng và phần mềm cơ sở dữ liệu; Tuy nhiên các thành phần ứng dụng và phần mềm cơ sở dữ liệu có thể có cùng thiết kế để vận hành một môi trường tập trung. Thực chất phần mềm mạng đã làm cho phần mềm ứng dụng và phần mềm cơ sở dữ liệu tưởng rằng chúng đang truy nhập cơ sở dữ liệu trong môi trường cục bộ. Một môi trường như vậy có thể phức tạp hơn mô hình tập trung bởi vì phần mềm mạng có thể phải thực hiện cơ chế đồng thời cho phép nhiều người dùng cuối có thể truy nhập vào cùng cơ sở dữ liệu.

c) Mô hình xử lý từng phần cơ sở dữ liệu (Database extract processing model):

Một mô hình khác trong đó một cơ sở dữ liệu ở xa có thể được truy nhập bởi phần mềm cơ sở dữ liệu, được gọi là xử lý dữ liệu từng phần.

Với mô hình này, người sử dụng có thể tại một máy tính cá nhân kết nối với hệ thống máy tính ở xa nơi có dữ liệu mong muốn. Người sử dụng sau đó có thể tác động trực tiếp đến phần mềm chạy trên máy ở xa và tạo yêu cầu để lấy dữ liệu từ cơ sở dữ liệu đó. Người sử dụng cũng có thể chuyển dữ liệu từ máy tính ở xa về chính máy tính của mình, vào đĩa cứng và có thể thực hiện việc sao chép bằng phần mềm cơ sở dữ liệu trên máy cá nhân.

Với cách tiếp cận này, người sử dụng phải biết chắc chắn là dữ liệu nằm ở đâu và làm như thế nào để truy nhập và lấy dữ liệu từ một máy tính ở xa. Phần mềm ứng dụng đi kèm cần phải có trên cả hai hệ thống máy tính để kiểm soát sự truy nhập dữ liệu và chuyển dữ liệu giữa hai hệ thống. Tuy nhiên, phần mềm cơ sở dữ liệu chạy trên hai máy không cần biết rằng việc xử lý cơ sở dữ liệu từ xa đang diễn ra vì người sử dụng tác động tới chúng một cách độc lập.

d) Mô hình cơ sở dữ liệu Client/Server (Client/Server database model):

Trong mô hình cơ sở dữ liệu Client/Server, cơ sở dữ liệu nằm trên một máy khác với các máy có thành phần xử lý ứng dụng. Nhưng phần mềm cơ sở dữ liệu được tách ra giữa hệ thống Client chạy các chương trình ứng dụng và hệ thống Server lưu trữ cơ sở dữ liệu.

Trong mô hình này, các thành phần xử lý ứng dụng trên hệ thống Client đưa ra yêu cầu cho phần mềm cơ sở dữ liệu trên máy client, phần mềm này sẽ kết nối với phần mềm cơ sở dữ liệu chạy trên Server. Phần mềm cơ sở dữ liệu trên Server sẽ truy nhập vào

cơ sở dữ liệu và gửi trả kết quả cho máy Client.

Mới nhìn, mô hình cơ sở dữ liệu Client/Server có vẻ giống như mô hình file - server, tuy nhiên mô hình Client/Server có rất nhiều thuận lợi hơn mô hình file - server. Với mô hình file - server, thông tin gắn với sự truy nhập cơ sở dữ liệu vật lý phải chạy trên toàn mạng. Một giao tác yêu cầu nhiều sự truy nhập dữ liệu có thể gây ra tắc nghẽn lưu lượng truyền trên mạng.

Giả sử một người dùng cuối tạo ra một vấn tin để lấy dữ liệu tổng số, yêu cầu đòi hỏi lấy dữ liệu từ 1000 bản ghi. Với cách tiếp cận file - server nội dung của tất cả 1000 bản ghi phải đưa lên mạng, vì phần mềm cơ sở dữ liệu chạy trên máy của người sử dụng phải truy nhập từng bản ghi để thỏa mãn yêu cầu của người sử dụng. Với cách tiếp cận cơ sở dữ liệu Client/Server, chỉ có lời vấn tin khởi động ban đầu và kết quả cuối cùng cần đưa lên mạng, phần mềm cơ sở dữ liệu chạy trên máy lưu giữ cơ sở dữ liệu sẽ truy nhập các bản ghi cần thiết, xử lý chúng và gọi các thủ tục cần thiết để đưa ra kết quả cuối cùng.

Front-end software

Trong mô hình cơ sở dữ liệu Client/Server, thường nói đến các phần mềm front-end software và back-end software. Front-end software được chạy trên một máy tính cá nhân hoặc một workstation và đáp ứng các yêu cầu đơn lẻ riêng biệt. Phần mềm này đóng vai trò của Client trong ứng dụng cơ sở dữ liệu Client/Server và thực hiện các chức năng hướng tới nhu cầu của người dùng cuối cùng, phần mềm Front-end software thường được chia thành các loại sau:

- *End user database software:* Phần mềm cơ sở dữ liệu này có thể được thực hiện bởi người sử dụng cuối trên chính hệ thống của họ để truy nhập các cơ sở dữ liệu cục bộ nhỏ cũng như kết nối với các cơ sở dữ liệu lớn hơn trên cơ sở dữ liệu Server.
- *Simple query and reporting software:* Phần mềm này được

thiết kế để cung cấp các công cụ dễ dùng hơn trong việc lấy dữ liệu từ cơ sở dữ liệu và tạo các báo cáo đơn giản từ dữ liệu đã có.

- *Data analysis software:* Phần mềm này cung cấp các hàm về tìm kiếm, khôi phục, chúng có thể cung cấp các phân tích phức tạp cho người dùng.
- *Application development tools:* Các công cụ này cung cấp các khả năng về ngôn ngữ mà các nhân viên hệ thống thông tin chuyên nghiệp sử dụng để xây dựng các ứng dụng cơ sở dữ liệu của họ. Các công cụ ở đây bao gồm các công cụ về thông dịch, biên dịch đơn đến các công cụ CASE (*Computer Aided Software Engineering*), chúng tự động tất cả các bước trong quá trình phát triển ứng dụng và sinh ra chương trình cho các ứng dụng cơ sở dữ liệu.
- *Database administration Tools:* Các công cụ này cho phép người quản trị cơ sở dữ liệu sử dụng máy tính cá nhân hoặc trạm làm việc để thực hiện việc quản trị cơ sở dữ liệu như định nghĩa các cơ sở dữ liệu, thực hiện lưu trữ hay phục hồi.

Back-end software

Phần mềm này bao gồm phần mềm cơ sở dữ liệu Client/Server và phần mềm mạng chạy trên máy đóng vai trò là Server cơ sở dữ liệu.

d) Mô hình cơ sở dữ liệu phân tán (Distributed database model):

Cả hai mô hình File - Server và Client/Server đều giả định là dữ liệu nằm trên một bộ xử lý và chương trình ứng dụng truy nhập dữ liệu nằm trên một bộ xử lý khác, còn mô hình cơ sở dữ liệu phân tán lại giả định bản thân cơ sở dữ liệu có ở trên nhiều máy khác nhau.

MỤC LỤC

Lời nói đầu 3

BÀI 1: GIỚI THIỆU TỔNG QUAN VỀ NGÔN NGỮ SQL SERVER

1. SQL là gì? 5
2. SQL là một chuẩn 5
3. Bảng CSDL 6
4. Câu truy vấn SQL 6
5. Cách tạo một bảng trong SQL 9
6. Cách nhập dữ liệu vào bảng 9

BÀI 2: TRUY XUẤT DỮ LIỆU

1. Truy xuất một cột 11
2. Truy xuất nhiều cột 11
3. Truy xuất tất cả các cột 12

BÀI 3: SẮP XẾP DỮ LIỆU

1. Sắp xếp một cột 14
2. Sắp xếp theo thứ tự cột 15

BÀI 4: LỌC DỮ LIỆU

BÀI 5: TRÌNH LỌC DỮ LIỆU CAO CẤP

BÀI 6: BỘ LỌC THÔNG MINH

1. Kí tự đại diện (%), cú pháp 26
2. Kí tự đại diện (_) 27
3. Kí tự đại diện khoảng trống ([]) 29

BÀI 7: TẠO CẤU TRÚC CỘT DỮ LIỆU

BÀI 8: SỬ DỤNG HÀM DỮ LIỆU

1. Hàm Upper 36
2. Hàm Soundex 37
3. Hàm ngày và thời gian 38

4. Hàm số 39

BÀI 9: TỔNG DỮ LIỆU

1. Hàm trung bình 41
2. Hàm count () 41
3. Hàm max () 42
4. Hàm min () 42
5. Hàm sum () 43
6. Hàm tính trung bình 43

BÀI 10: TẠO NHÓM DỮ LIỆU

1. Hiểu về tạo nhóm 45
2. Tạo nhóm 45
3. Bộ lọc nhóm 46
4. Phân biệt hai mệnh đề Group by và Order by 48

BÀI 11: LÀM VIỆC TỪ TRUY VẤN

1. Lọc dữ liệu với từ truy vấn 50
2. Dùng Subquery như một trường tính toán 52

BÀI 12: LIÊN KẾT BẢNG

1. Tạo một liên kết 54
2. Tạo liên kết nội bộ (Inner Joins) 59
3. Liên kết nhiều bảng 60

BÀI 13: TẠO LIÊN KẾT BẢNG THÔNG MINH

1. Sử dụng các định danh cho bảng 62
2. Các kiểu liên kết 64
3. Sử dụng liên kết với các hàm trung bình 68

BÀI 14: NỐI CÁC CÂU TRUY VẤN

BÀI 15: CHÈN DỮ LIỆU

1. Chèn vào một dòng đã có sẵn 73
2. Chèn vào từng phần của dòng 76
3. Chèn kết quả dữ liệu truy vấn vào bảng 77
4. Chép từ một bảng tới bảng khác 79

BÀI 16: CẬP NHẬT VÀ XÓA DỮ LIỆU
1. Cập nhật dữ liệu 80
2. Xóa dữ liệu 81
BÀI 17: TẠO VÀ CHỈNH SỬA BẢNG
1. Tạo bảng 82
2. Làm việc với giá trị rỗng (null) 83
3. Giá trị mặc định (default) 83
4. Cập nhật bảng 85
5. Xóa bảng 86
6. Đổi tên bảng 86
BÀI 18: SỬ DỤNG VIEWS
1. Hiểu rõ về views 87
2. Một số lý do để bạn sử dụng views 88
3. Một số nguyên tắc và giới hạn của views 88
4. Tạo views 89
BÀI 19: LÀM VIỆC VỚI THỦ TỤC LƯU TRỮ
BÀI 20: KIỂM SOÁT VIỆC XỬ LÝ GIAO TÁC
1. Hiểu về xử lý giao tác 102
2. Điều khiển các giao tác 104
BÀI 21: SỬ DỤNG CON TRỎ
1. Làm việc với con trỏ 108
2. Tạo con trỏ 108
3. Sử dụng con trỏ 109
4. Đóng con trỏ 111
BÀI 22: TÌM HIỂU MỘT SỐ ĐẶC TRƯNG CỦA SQL
1. Tìm hiểu về ràng buộc 112
2. Tìm hiểu về Indexes 115
3. Tìm hiểu về Triggers 116

PHỤ LỤC A

1. Bảng Vendors (đại lý) 117
2. Bảng Products (sản phẩm) 123
3. Bảng Customers (khách hàng) 130
4. Bảng Custnew (khách hàng mới) 136
5. Bảng Orders (hóa đơn) 139
6. Bảng OrderItems 142

PHỤ LỤC B: DIỄN GIẢI CÁC BẢNG VÍ DỤ

1. Bảng Vendors (đại lý) 153
2. Bảng Products (sản phẩm) 153
3. Bảng Customers (khách hàng) 154
4. Bảng Orders (hóa đơn) 154
5. Bảng OrderItems (chi tiết hóa đơn) 155

PHỤ LỤC C: CÁC KIỂU DỮ LIỆU TRONG SQL

1. Kiểu chuỗi (String) 156
2. Kiểu số (Numeric) 156
3. Kiểu ngày giờ (Date và time) 157
4. Kiểu nhị phân (Binary) 157

PHỤ LỤC D: MỘT SỐ TRÌNH ỨNG DỤNG CỦA SQL

1. Sử dụng JRUN 3.X 158
2. Sử dụng Microsoft Access 159
3. Sử dụng Microsoft ASP 162
4. Sử dụng Microsoft SQL Server 2000 163
5. Sử dụng Microsoft Visual Basic 163
6. Sử dụng Oracle9i 164
7. Sử dụng SQLPRAC để thực hành các câu lệnh SQL 164

PHỤ LỤC E: MỘT SỐ VẤN ĐỀ CỦA SQL

1. SQL – Công cụ quản lý CSDL điển hình 166
1. Oracle 9i: Đơn giản hóa cú pháp SQL 171
3. Các cấu hình cơ sở dữ liệu Client/Server 176



Nơi phát hành

Công Ty TNHH BÚT

Nhà sách VĂN N

172 Đinh Tiên Hoàng, Q.

ĐT: 8201686 - 8206279

Email: v-nghe@hcm.fpt.vn

tự học sql (có đĩa)

1 C04080 500176

29.000 VND

Giá : 29.000 đ

(Bao gồm cả đĩa C