SQL- Những kiến thức cơ bản

GIỚI THIỆU

Ngôn ngữ truy vấn có cấu trúc (SQL) là một ngôn cơ sở dữ liệu (CSDL) chuẩn công nghiệp được công cụ quản trị dữ liệu của Microsoft (Microsoft jet database engine) sử dụng. SQL được sử dụng để tạo những đối tượng truy vấn (QueryDef objects), như là đối số cho phương thức mở tập hợp bản ghi (OpenRecordset method), và là thuộc tính nguồn bản ghi (RecordSource property) của điều khiển dữ liệu (data control). Nó cũng có thể được dùng với những phương thức thi hành (Execute method) để trực tiếp tạo và thao tác ... (jet databases), và tạo ra các SQL PassThrough truy vấn để thao tác trên các CSDL khách chủ từ xa (remote client/server databases).

Chương này sẽ bàn tới cấu trúc cơ bản của SQL, và cách thức sử dụng nó cho việc tạo, bảo trì và sửa đổi CSDL. Chúng ta cũng nói tới sự xây dựng và công dụng của truy vấn SQL để tạo các đối tượng tập hợp bản ghi (Recordset objects), và để chọn, sắp xếp, lọc và cập nhật dữ liệu trong những bảng cơ sở. Hơn nữa, chương này sẽ xem xét cách thức tối ưu hoá truy vấn SQL về mặt tốc độ và hiệu quả. Cuối cùng, chúng ta bàn tới sự khác nhau giữa Microsoft Jet SQL và ANSI SQL một cách cụ thể

SQL LÀ GÌ ?

SQL là một ngôn ngữ lập trình về CSDL có nguồn gốc liên quan mật thiết tới sự phát minh ra mô hình CSDL quan hệ của E.F.Codd vào đầu những năm 70. Tiền thân của SQL là ngôn ngữ Sequel, và vì lý do này SQL vẫn thường được phát âm là “sequel” hơn là “ess cue ell”, mặc dầu cả hai cách phát âm đều được chấp nhận.

SQL ngày nay phát triển rộng và trở thành một ngôn ngữ chuẩn cho CSDL quan hệ, và đã được định nghĩa bởi chuẩn ANSI. Hầu hết các bản thi hành của SQL chỉ là sự biến đổi nhỏ từ SQL chuẩn, bao gồm cả phiên bản được Jet database engine hỗ trợ. Những sự khác nhau này sẽ được nhắc tới ở cuối chương, nhưng hầu hết các cấu trúc và các chức năng của ngôn ngữ là nhất quán đối với các nhà phát triển các hệ quản trị CSDL. Nếu bạn đã sử dụng bất cứ bản thi hành nào của SQL, bạn sẽ thấy không khó khăn mấy khi chuyển sang Microsoft Jet SQL.

SQL vs. Navigation

Như đã đề cập trong phần đầu tài liệu, Mircosoft Jet database engine cung cấp hai phương thức tách biệt để hoàn tất hầu hết các tác vụ CSDL:

- Một mô hình điều hướng dựa trên cở sở dịch chuyển qua lại giữa các bản ghi.
- Một mô hình quan hệ dựa trên truy vấn hỏi có cấu trúc (SQL).

Mô hình điều hướng bao gồm những thuộc tính và phương thức được mô tả trong “Tạo và sửa đổi CSDL” (“Creating and Modifying Databases”) và “Thao tác với bản ghi và trường” (“Working with Records and Fields”). Mô hình quan hệ được nói bàn tới trong chương này.

Những lập trình viên không quen thuộc với những hệ quản trị cơ sở dữ liệu hướng file như dBASE, Foxpro, và Paradox có thể cảm thấy dễ chịu khi bắt đầu với các phương thức điều hướng được thảo luận trong chương trước. Tuy nhiên, trong hầu hết các trường hợp những phương thức SQL với vai trò tương đương tỏ ra hiệu quả hơn, và nói

chung chúng nên được dùng cho những nơi tính hiệu quả được xem là quan trọng hơn cả. Hơn nữa SQL có một lợi điểm là một giao tiếp ở mức chuẩn công nghiệp về CSDL, thế nên một sự hiểu biết về các lệnh SQL cho phep bạn truy cập và thao tác với một diện rộng các sản phẩm CSDL từ các nhà phát triển khác nhau.

CÁC THÀNH PHẦN CỦA SQL

Ngôn ngữ SQL bao gồm các lệnh, các mệnh đề, các toán tử, và các hàm tổng hợp (hàm nhóm - aggregate functions). Những thành phần này được kết hợp vào trong các phát biểu (statements) dùng để tạo, cập nhật, và thao tác trên CSDL. Những mục sau sẽ mô tả những thành tố đó một cách ngắn gọn, và phần còn lại của chương này sẽ đưa ra cho bạn những ví dụ cụ thể về công dụng của chúng.

Chú ý: Những mục sau sẽ những lệnh và từ khoá được dùng thường xuyên nhất, nhưng không phải tất cả. Để có một tham khảo hoàn chỉnh về danh sách các từ khoá SQL, hãy tìm kiếm "SQL" trong Books Online.

1.Lệnh SQL:

Giống như mô hình điều hướng của DAO (Data Access Object), SQL cung cấp cả hai phần, ngôn ngữ định nghĩa dữ liệu (DDL - Data Definition Language) và ngôn ngữ thao tác dữ liệu (DML - Data Manipulation Language). Tuy có vài phần trùng lặp, nhưng những câu lệnh DDL cho phép bạn tạo và định nghĩa các CSDL, các trường, các chỉ mục mới, trong khi những câu lệnh DML để bạn xây dựng các truy vấn, sắp xếp, lọc, và trích dữ liệu từ trong CSDL.

DDL

Các câu lệnh DDL trong SQL là biểu thức được xây dựng chung quanh những mệnh đề sau:

CREATE Dùng để tạo mới các bảng, các trường và các chỉ mục.
DROP Dùng để xoá các bảng và chỉ mục khỏi CSDL.
ALTER Dùng để sửa đổi các bảng bằng cách thêm trường, thay đổi định nghĩa của các trường.

DML

Các câu lệnh DML là các biểu thức được xây dựng dựa trên các mệnh đề sau:

SELECT Dùng để truy vấn CSDL để lấy được những bản ghi thoả mãn những tiêu chuẩn nào đó.
INSERT Dùng để chèn một nhóm dữ liệu vào CSDL thông qua một thao tác.
UPDATE Dùng để thay đổi giá trị của những trường, những bản ghi cụ thể.
DELETE Dùng để loại bỏ những bản ghi ra khỏi CSDL.

2.Mệnh đề SQL:

Mệnh đề là những điều kiện thay đổi được dùng để xác định dữ liệu bạn muốn chọn, muốn thao tác. Bảng sau liệt kê những mệnh đề bạn có thể dùng.

FROM Liệt kê danh sách các bảng mà ta cần lấy các bản ghi từ đó.
WHERE Xác định các điều kiện mà bản ghi được chọn phải đáp ứng được.
GROUP BY Dùng để nhóm các bản ghi được chọn thành các nhóm riêng biệt.
HAVING Dùng để đưa ra điều kiện cho mỗi nhóm.
ORDER BY Dùng để sắp xếp các bản ghi được chọ theo một thứ tự nào đó.

3.Những toán hạng SQL:

Có hai loại toán hạng trong SQL: toán hạng logic và toán hạng so sánh.

Toán hạng logic:
Toán hạng logic được dùng để nối các biểu thức, thường là trong phạm vi của mệnh đề WHERE. Ví dụ như:

SELECT * FROM MY_TABLE WHERE Condition1 AND Condition2;

Những toán tử logic bao gồm: AND, OR, NOT

Toán hạng so sánh:

Toán hạng so sánh được dùng để so sánh tương đối giá trị hai biểu thức để xác định những hoạt động nào sẽ được thực hiện. Ví dụ:

SELECT * FROM Publishers WHERE PubID = 5;

Những toán tử so sánh bao gồm:

< bé hơn
<= bé hơn hoặc bằng
> lớn hơn
>= lớn hơn hoặc bằng
= bằng
<> khác

4.Hàm tổng hợp

(aggregate functions)

Hàm tổng hợp (hàm nhóm) được dùng trong phạm vi của mệnh đề SELECT trên một nhóm bản ghi để trả lại một giá trị. Ví dụ, hàm AVG có thể trả lại giá trị trung bình của tất cả các giá trị trong một trường cụ thể. Bảng sau liệt kê danh sách các hàm tổng hợp.

AVG Trả lại giá trị trung bình trong một trường. COUNT Trả lại số bản ghi được chọn.
SUM Hàm tính tổng các giá trị trong một trường cụ thể.
MAX Hàm trả về giá trị cực đại của trường đó.
MIN Hàm trả về giá trị cực tiểu của trường đó

NHỮNG THAO TÁC DLL

DDL bao gồm một số lệnh bạn có thể dùng để tạo bảng và chỉ mục, và sửa đổi các bảng bằng cách thêm hoặc loại bỏ các cột hoặc chỉ mục. Những câu lệnh định nghĩa dữ liệu có thể chỉ được dùng với Jet database; Chúng không được hỗ trợ cho bất cứ CSDL định dạng ngoài.

Chú ý: Để dùng câu lệnh DDL, hoặc bất cứ truy vấn nào không trả lại tập bản ghi, hãy đóng ngoặc kép và sử dụng chúng như là đối số của các phương thức thi hành của CSDL hay đối tượng truy vấn (QueryDef object) như trong ví dụ sau:

MyDB.Execute "CREATE TABLE Employees ([First Name] TEXT, [Last_ Name] TEXT)";

Để dùng bất cứ một câu lệnh nào trả lại các bản ghi (như SELECT), dùng biểu thức như là đối số nguồn của phương thức mở tập bản ghi (OpenRecordset method), như trong ví dụ sau:

MyDB.OpenRecordset ("SELECT * FROM Titles WHERE Au_ID = 5", _ dbOpenDynaset);

1.Tạo một bảng:

Để tạo một bảng trong CSDL, dùng câu lệnh CREATE TABLE. Một câu lệnh hoàn chỉnh nhận các đối số là tên bảng, tên các trường, kiểu dữ liệu của các trường và độ rộng của các trường.

Ví dụ sau tạo một bảng có tên là "Employees", có hai trường kiểu TEXT với độ rộng là 25:

CREATE TABLE Employees ([First Name] TEXT(25), [Last Name] TEXT(25));

Thêm và xoá cột:

Bạn có thể thêm, sửa đổi hoặc xoá các cột với câu lệnh ALTER TABLE. Ví dụ, câu lệnh sau thêm một trường kiểu TEXT có độ rộng 25 và tên là "Notes" vào bảng Employees:

ALTER TABLE Employees ADD COLUMN Notes TEXT(25);

Để loại bỏ một cột, dùng từ khoá DROP. Ví dụ này loại bỏ cột có tên là "Notes" mới vừa được thêm lúc nãy:

ALTER TABLE Employees DROP COLUMN Notes;

Để sửa đổi một trường, trước tiên bạn phải xoá nó, và sau đó là thêm trường mới với tên như cũ. Ví dụ sau tăng độ rộng của trường "Notes":

ALTER TABLE Employees DROP COLUMN Notes;
ALTER TABLE Employees ADD COLUMN Notes TEXT(30);
Chú ý: Dùng ALTER TABLE, bạn chỉ có thể thêm hoặc xoá một trường tại mỗi thời điểm.

2.Tạo và xoá chỉ mục:

Có ba cách khác nhau để tạo chỉ mục:

- Lúc bắt đầu tạo bảng với câu lệnh CREATE TABLE
- Với câu lệnh CREATE INDEX.
- Với câu lệnh ALTER TABLE

Mặc dầu cả ba cách này đều cho kết quả tương tự, nhưng vẫn có những khác điểm khác nhau. Nếu bạn muốn thêm một khoá ngoại (foreign key) và ép buộc toàn vẹn tham chiếu (enfore referential integrity), bạn phải dùng một mệnh đề ràng buộc (CONSTRAINT clause) trong các câu lệnh CREATE TABLE hoặc ALTER TABLE.

Đôi khi người ta muốn tạo một bảng ban đầu không có chỉ mục, và tiếp đó là thiết kế các tham số chỉ mục sau khi dùng mẫu bảng. Với tình huống này, bạn nên dùng CREATE TABLE để tạo mẫu bảng không có chỉ mục, và sau đó thêm các chỉ mục với câu lệnh CREATE INDEX hoặc ALTER TABLE.

Tạo một chỉ mục với câu lệnh CREATE TABLE.

Khi bạn tạo một bảng, bạn có thể tạo một chỉ mục cho từng cột riêng rẽ, hoặc hai hoặc nhiều hơn các cột, dùng mệnh đề SQL CONSTRAINT (từ khoá CONSTRAINT bắt đầu định nghĩa một chỉ số). Ví dụ sau đây minh hoạ cách tạo ra một bảng với ba trường có chỉ mục:

CREATE TABLE Employees ([First Name] TEXT(25), [Last Name] TEXT(25), _ [Date of Birth] DATETIME, CONSTRAINT EmployeesIndex UNIQUE _ ([First Name], [Last Name], [Date of Birth]));

Để đánh chỉ mục với một cột, bạn đặt mệnh đề CONSTRAINT vào một trong những mô tả cột. Ví dụ, để đánh chỉ mục trường "Date of Birth", bạn dùng câu lệnh CREATE TABLE sau đây:

CREATE TABLE Employees ([First Name] TEXT(25), [Last Name] TEXT(25), _ [Date of Birth] DATETIME CONSTRAINT EmployeesIndex PRIMARY);

Sự khác nhau giữa đánh chỉ mục cho nhiều trường và cho một trường là: cho một trường đơn, từ khoá CONSTRAINT bắt đầu định nghĩa chỉ mục không bị tách biệt với trường cuối cùng bởi dấu phẩy mà đi sát ngay sau kiểu dữ liệu của trường được đánh chỉ mục đó.

Tạo chỉ mục với câu lệnh CREATE INDEX.

Bạn cũng có thể dùng mệnh đề CREATE INDEX để thêm một chỉ mục. Ví dụ sau đây đưa ra cùng một kết quả với ví dụ trước, ngoại trừ việc dùng CREATE TABLE thay thế cho ALTER TABLE.

CREATE UNIQUE INDEX MyIndex ON Employees ([Date of Birth]);

Trong mệnh đề tuỳ chọn WITH, bạn có thể ép buộc dữ liệu với ràng buộc PRIMARY, có nghĩa đây là trường chỉ mục chính; DISALLOW NUL, nghĩa là trường này không bị bỏ trống; IGNORE NULL, có nghĩa bản ghi đó sẽ không được đánh chỉ mục nếu trường đó để trống.

Ví dụ sau thêm mệnh đề WITH vào ví dụ trước, để cho không bản ghi nào có thể được thêm vào bảng mà cột số bảo hiểm xã hội bị bỏ trống:

CREATE UNIQUE INDEX MyIndex ON Employees (SSN) _ WITH DISALLOW NULL;

Chú ý: Không dùng từ khoá PRIMARY khi bạn tạo ra một chỉ mục mới trong bảng mà bảng đó đã tồn tại khoá chính (Primary key); Nếu bạn vi phạm điều này thì hệ thống sẽ báo lỗi.

Bạn đang dùng CREATE INDEX để tạo một đặc tả chỉ mục trên một bảng mà chưa tồn tại chỉ mục nào. Để tạo một chỉ mục như thế; bạn không cần sự cho phép hoặc truy cập tới một máy chủ ở xa, và CSDL ở xa không nhận biết được hay không hề ảnh hưởng bởi chỉ mục đó. Bạn dùng cùng một cú pháp cho bảng kết nối và bảng gốc. Điều này đặc hữu dụng khi tạo một chỉ mục trên một bảng thường là chỉ đọc (read only) bởi vì nó thiếu một chỉ mục.

Tạo một chỉ mục với câu lệnh ALTER TABLE

Bạn cũng có thể thêm một chỉ mục cho một bảng đã tồn tại bằng cách dùng câu lệnh ALTER TABLE, dùng cú pháp ADD CONSTRAINT. Ví dụ sau thêm một chỉ mục cho trường "SSN":

ALTER TABLE Employees ADD CONSTRAINT MyIndex _ PRIMARY (SSN);

Bạn cũng có thể thêm chỉ mục cho nhiều trường và một bảng bằng cách dùng câu lệnh ALTER TABLE giống như sau:

ALTER TABLE Employees ADD CONSTRAINT NameIndex _ UNIQUE ([Last Name], [First Name], SSN);

Mệnh đề CONSTRAINT và toàn vẹn tham chiếu (Referential Integrity).

Một ràng buộc là một chỉ mục. Bạn dùng mệnh đề CONSTRAINT để tạo hoặc xoá các chỉ mục với các câu lệnh CREATE TABLE và ALTER TABLE, như đã chỉ ra ở phần trước.

Mệnh đề CONSTRAINT cũng cho phép bạn định nghĩa khoá chính và khoá ngoại, định nghĩa các quan hệ và ép buộc toàn vẹn tham chiếu.

Để biết thêm thông tin về quan hệ và toàn vẹn tham chiếu, hãy xem cuốn "Tạo và sửa đổi CSDL" ("Creating and Modifying Databases").

Có hai loại mệnh đề CONSTRAINT: Một để tạo chỉ mục cho từng trường đơn và một để tạo chỉ mục cho nhiều hơn một trường.

Cú pháp của chỉ mục trên một trường là:

CONSTRAINT name {PRIMARY KEY | UNIQUE | REFERENCES foreigntable [(foreginfield1, foreignfield2)]}

Cú pháp cho chỉ mục trên nhiều trường là:

CONSTRAINT name {PRIMARY KEY (primary1[,primary2[,...]]) | UNIQUE (unique1[,unique2[,...]]) | FOREIGN KEY (ref1[,ref2[,...]]) REFERENCES foreigntable [(foreignfield1[,foreignfield2[,...]])};

Sau đây là các đối số áp dụng cho hai loại trên:

name : Tên của chỉ mục được tạo.
primary1, primary2 : Tên của trương hay các trường được chỉ định làm khoá chính.
unique1, unique2 : Tên của trường hay các trường được chỉ định làm khoá không lặp.
ref1, ref2 : Tên của trường hoặc các trường khoá ngoài tham chiếu tới một trường, một số trường ở bảng khác.
foreigntable Tên của bảng ngoài chứa một hoặc một số trường được xác định bởi foreignfield.
foreignfield1, foreignfield2: Tên của trường hoặc một số trường trong bảng ngoài được xác định bởi ref1, ref2.

Dùng CONSTRAINT, bạn có thể gán cho một trường như một trong những loại chỉ mục sau:

- UNIQUE – Chỉ định trường một trường có giá trị không lặp. Điều này có nghĩa là hai bản

ghi bất kỳ trong bảng không có cùng giá trị trong trường này. Bạn có thể ràng buộc bất kỳ trường nào hoặc một danh sách các trường là duy duy nhất (unique). Nếu nhiều trường được chỉ định là không lặp, bộ giá trị kết hợp của các trường đó phải là duy nhất, dầu là hai hoặc một số bản ghi có cùng giá trị trong một trường của nhóm các trường đó.

- PRIMARY KEY – Chỉ định một hoặc một tập các trường trong bảng tạo thành khoá chính. Tất cả giá trị trong khoá chính phải duy nhất, và có một khoá chính duy nhất cho một bảng. Nếu bạn thiết lập một khoá chính cho một bảng đã tồn tại khoá chính thì hệ thống sẽ báo lỗi.

- FOREIGN KEY – Xác định một trường như một khoá ngoài. Nếu khoá chính của bảng ngoài có nhiều hơn một trường, bạn phải dùng một định nghĩa cho chỉ mục nhiều trường, liệt kê tất cả các trường tham chiếu, tên của các bảng, tên của bảng ngoài, và tên của các trường được tham chiếu trong bảng ngoài theo cùng một thứ tự như đã liệt kê danh sách các trường tham chiếu. Nếu trường được tham chiếu là khoá chính của bảng ngoài, bạn không cần chỉ định trường được tham chiếu mà Jet engine đã ngầm định khoá chính cúa bảng ngoài là trường được tham chiếu.

Ví dụ, để thêm một chỉ mục cho bảng Titles trong CSDL Biblio.mdb, bạn có thể dùng câu lệnh sau đây:

ALTER TABLE Titles ADD CONSTRAINT MyIndex _ FOREIGN KEY (PubID) REFERENCES Publishers (PubID);

Nhớ rằng, bằng cách dùng từ khoá FOREIGN KEY, Chúng ta đang thiết lập một quan hệ giữa trường PubID của bảng Titles (khoá ngoài) và trường PubID trong bảng Publishers (khoá chính). Mối quan hệ này sẽ được ràng buộc bởi Jet engine, như thể bạn đang dùng phương thức CreateRelation được mô tả trong "Tạo và sửa đổi CSDL"

PHẦN NGÔN NGỮ THAO TÁC TRÊN DỮ LIỆU - DML

Phần ngôn ngữ thao tác trên dữ liệu (DML - Data Manipulation Languague) được dùng để lấy các bản ghi trong các bảng, cập nhật, thêm, xoá các bản ghi của các bảng. Có một số câu lệnh hỗ trợ các tác vụ này, nhưng phần lớn là có cấu trúc của câu lệnh SELECT.

Truy vấn chọn:

Sử dụng câu lệnh SELECT để lấy các bản ghi từ CSDL như một tập hợp các bản ghi, lưu trử chúng trong một đối tượng tập bản ghi mới (Recordset object). ứng dụng của bạn có thể thao tác trên tập bản ghi này như hiển thị, thêm, thay đổi và xoá nếu cần thiết. ứng dụng của bạn cũng có thể hiển thị, sinh các báo cáo từ dữ liệu đó.

SELECT thường là từ đầu tiên trong một câu lệnh SQL. Hầu hết các câu lệnh hoặc là SELECT hoặc là SELECT...INTO. Bạn có thể dùng một câu lệnh SELECT trong SQL là thuộc tính của đối tượng truy vấn (QueryDef object), là thuộc tính RecordSource của một điều khiển dữ liệu (data control), hoặc một đối số cho phương thức OpenRecordset. câu lệnh SELECT không thay đổi dữ liệu trong CSDL; chúng chỉ lấy dữ liệu ra từ CSDL.

Dạng tổng quát của câu lệnh SELECT là:

SELECT fieldlist
FROM tablenames IN databasename
WHETE searchconditions

GROUP BY fieldlist
HAVING group criteria
ORDER BY fieldlist
WITH OWNERACCESS OPTION

Mỗi phần trong câu lệnh đại diện cho một mệnh đề được bàn đến ở các phần sau:

Truy vấn đơn giản:
Dạng đơn giản nhất của câu lệnh SELECT là:
SELECT * FROM tablename;

Ví dụ, truy vấn chọn sau trả lại tất cả các cột của tất cả các bản ghi trong bảng Employees:
SELECT * FROM Employees;

Dấu sao cho biết rằng tất cả các trường của bảng được chọn. Bạn cũng có thể chỉ định một số trường nhất định. Khi hiển thị, dữ liệu trong mỗi cột sẽ hiển theo thứ tự như chúng đã được liệt kê, vì vậy bạn có thể thay đổi lại thứ tự cho dễ đọc:

SELECT [First Name], [Last Name] FROM Employees;

Chỉ định nguồn dữ liệu được chọn:

Một câu lệnh SELECT luôn có mệnh đề FROM, cho biết danh sách các bảng ta cần lấy các bản ghi từ đó.

Nếu một trường tồn tại trong nhiều bảng trong mệnh đề FROM, đặt trước chúng tên trường và dấu chấm. Trong ví dụ sau, trường Department có trong cả hai bảng Employees và Supervisors. Câu lệnh chỉ chọn trường Department của bảng Employees và SupvName từ bảng Supervisors:

SELECT Employees. Department, SupvName _ FROM Employees, Supervisors _ WHERE Employees.Department = Supervisors.Department;

Khi mệnh đề FROM liệt kê nhiều hơn một bảng, thứ tự của chúng không quan trọng.

Xác định một bảng từ một CSDL bên ngoài.
Đôi khi, bạn cần thiết tham chiếu tới một bảng của một CSDL bên ngoài mà công cụ quản trị CSDL (Microsoft Jet database engine) có thể kết nối tới, như CSDL dBASE, Paradox hoặc một Jet database bên ngoài. Bạn có thể làm điều này bằng mệnh đề tuỳ chọn IN. Mệnh đề IN thường xuất hiện sau tên bảng trong mệnh đề FROM, nhưng cũng có thể được dùng trong SELECT INTO hoặc INSERT INTO, khi đích là một CSDL ngoài.

Chú ý: Bạn chỉ có thể IN để kết nối một CSDL ngoài tại một thời điểm.

Trong một số trường hợp, đối số đường dẫn đề cập tới cả thư mục chứa CSDL. Ví dụ, khi làm việc với dBASE, Foxpro, hoặc Paradox, tham số đường dẫn chỉ ra các thư mục chứa các file có đuôi .DBF hoặc .DB. Tên bảng được bắt nguồn từ đích hoặc biểu thức bảng.

Để xác định không phải là một Jet database, thêm dấu chấm phẩy và sau tên, và đóng lại bằng dấu trích đơn hoặc dấu ngoặc kép. Ví dụ:

'dBASE IV;'

Bạn cũng có thể dùng từ khoá DATABASE để chỉ định CSDL ngoài. Ví dụ, cả hai dòng

sau chỉ ra cùng một bảng;

SELECT * FROM Table IN "" [dBASE IV; _ DATABASE=C:\DBASE\DATA\SALES;];

SELECT * FROM Table IN "C:\DBASE\DATA\SALES" _ "dBASE IV;"

Chú ý: Để nâng hiệu quả và dễ sử dụng, thường người ta dùng bảng kết nối thay cho mệnh đề IN.

Để biết thêm thông tin về bảng kết nối, xem cuốn "Working with Records and Fields" và cuốn "Accessing External Data".

Biệt danh của cột.

Khi đối tượng Recordset được tạo ra từ câu lệnh SELECT, tên cột của bảng trở thành tên trường của đối tượng Recordset. Nếu bạn muốn tên khác đi, dùng mệnh đề AS. Ví dụ sau dùng "DOB" là biệt danh của trường [Date of Birth] trong bảng Employees: SELECT [Date of Birth] AS DOB FROM Employees;

Bất cứ khi nào bạn dùng truy vấn trả lại tên trường nhập nhằng hoặc trùng tên trường, bạn phải dùng mệnh đề AS để cung cấp tên khác nhau cho các trường. Ví dụ sau dùng bó danh "Head Count" để gán kết quả đếm trong tập bản ghi:

SELECT COUNT(EmployeeID) AS [Head Count] FROM Employees;

Sử dụng biến Visual Basic trong câu lệnh SQL

Trong một chương trình Visual Basic, bạn có thể tạo một câu lệnh SELECT trong ứng dụng của bạn bằng cách ghép các biến cục bộ vào trong một câu lệnh khi cần để chọn, sắp xếp, lọc dữ liệu được yêu cầu bởi ứng dụng của bạn. Ví dụ bạn có một điều khiển TextBox (TitleWanted) chứa tên của một tiêu đề và bạn muốn lấy tất cả các sách trong bảng Titles có nhan đề như nhan đề trong hộp TextBox, bạn có thể tạo một câu lệnh SQL bao gồm cả giá trị hiện thời của hộp TextBox. Nhưng nhớ rằng SQL đóng TitleWanted trong một dấu trích đơn ('):

Set Rst = Db.OpenRecordset("SELECT * FROM Titles " _ & "WHERE Title = '" & TitleWanted.Text & "'")

1.Lọc và sắp xếp kết quả của truy vấn:
SQL cung cấp một số từ khóa xác nhận và mệnh đề tuỳ chọn giúp bạn thuận tiện hơn trong việc hạn chế và sắp xếp kết quả. Phần sau sẽ thảo luận về sự tiện dụng này.

Chỉ dẫn DISTINCT

Để bỏ qua các bản ghi trùng nhau, dùng từ khoá DISTINCT. Nếu được dùng, giá trị trong trường hay một nhóm các trường được chọn trong câu lệnh SELECT sẽ là duy nhất. Ví dụ, Có một vài nhân viên liệt kê trong bảng Employees có cùng họ. Nếu hai bản ghi có cùng nội dung trường "Last Name" là Smith thì câu lệnh sau sẽ trả lại một bản ghi có nội dung là Smith:

SELECT DISTINCT [Last Name] FROM Employees;

Nếu bỏ từ khoá DISTINCT thì truy vấn sẽ trả lại nhiều hơn một giá trị Smith.

Kết quả tập hợp bản ghi của truy vấn dùng DISTINCT không cho phép cập nhật và không phản ánh được những thay đổi sau đó của người dùng khác.

Chỉ dẫn TOP

Để trả lại một số bản ghi nhất định ở đầu hoặc ở cuối của phạm vi các bản ghi. Dùng chỉ dẫn TOP. Giả sử bạn muốn lấy tên của 25 sinh viên đầu của lớp tốt nghiệp năm 1994:

SELECT TOP 25 [First Name], [Last Name] FROM Students _ WHERE [Graduation Year] = 1994 _ ORDER BY [Grade Point Average] DESC;

Nếu bạn không dùng mệnh đề ORDER BY, truy vấn sẽ trả lại 25 bản ghi tuỳ ý trong bảng Students thoả mãn điều kiện trong mệnh đề WHERE.

Bất cứ khi nào bạn dùng truy vấn trả lại tên trường nhập nhằng hoặc trùng tên trường, bạn phải dùng mệnh đề AS để cung cấp tên khác nhau cho các trường. Ví dụ sau dùng bó danh "Head Count" để gán kết quả đếm trong tập bản ghi:

SELECT COUNT(EmployeeID) AS [Head Count] FROM Employees;

Sử dụng biến Visual Basic trong câu lệnh SQL

Trong một chương trình Visual Basic, bạn có thể tạo một câu lệnh SELECT trong ứng dụng của bạn bằng cách ghép các biến cục bộ vào trong một câu lệnh khi cần để chọn, sắp xếp, lọc dữ liệu được yêu cầu bởi ứng dụng của bạn. Ví dụ bạn có một điều khiển TextBox (TitleWanted) chứa tên của một tiêu đề và bạn muốn lấy tất cả các sách trong bảng Titles có nhan đề như nhan đề trong hộp TextBox, bạn có thể tạo một câu lệnh SQL bao gồm cả giá trị hiện thời của hộp TextBox. Nhưng nhớ rằng SQL đóng TitleWanted trong một dấu trích đơn ('):

Set Rst = Db.OpenRecordset("SELECT * FROM Titles " _ & "WHERE Title = '" & TitleWanted.Text & "'")

1.Lọc và sắp xếp kết quả của truy vấn:
SQL cung cấp một số từ khóa xác nhận và mệnh đề tuỳ chọn giúp bạn thuận tiện hơn trong việc hạn chế và sắp xếp kết quả. Phần sau sẽ thảo luận về sự tiện dụng này.

Chỉ dẫn DISTINCT

Để bỏ qua các bản ghi trùng nhau, dùng từ khoá DISTINCT. Nếu được dùng, giá trị trong trường hay một nhóm các trường được chọn trong câu lệnh SELECT sẽ là duy nhất. Ví dụ, Có một vài nhân viên liệt kê trong bảng Employees có cùng họ. Nếu hai bản ghi có cùng nội dung trường "Last Name" là Smith thì câu lệnh sau sẽ trả lại một bản ghi có nội dung là Smith:

SELECT DISTINCT [Last Name] FROM Employees;

Nếu bỏ từ khoá DISTINCT thì truy vấn sẽ trả lại nhiều hơn một giá trị Smith.

Kết quả tập hợp bản ghi của truy vấn dùng DISTINCT không cho phép cập nhật và không phản ánh được những thay đổi sau đó của người dùng khác.

Chỉ dẫn TOP

Để trả lại một số bản ghi nhất định ở đầu hoặc ở cuối của phạm vi các bản ghi. Dùng chỉ dẫn TOP. Giả sử bạn muốn lấy tên của 25 sinh viên đầu của lớp tốt nghiệp năm 1994:

SELECT TOP 25 [First Name], [Last Name] FROM Students _ WHERE [Graduation Year] = 1994 _ ORDER BY [Grade Point Average] DESC;

Nếu bạn không dùng mệnh đề ORDER BY, truy vấn sẽ trả lại 25 bản ghi tuỳ ý trong bảng Students thoả mãn điều kiện trong mệnh đề WHERE.

Chỉ dẫn TOP không chọn lựa giữa những bản ghi bằng nhau. Trong ví dụ, nếu bản ghi thứ 25 và thứ 26 có cùng hạng thì truy vấn sẽ trả lại 26 bản ghi.

Bạn cũng có thể dùng từ khoá PERCENT để trả lại một số phần trăm bản ghi ở đầu hay cuối tuỳ thuộc vào mệnh đề ORDER BY. Giã sử rằng thay vì 25 sinh viên, bạn muốn 10 phần trăm sinh viên của lớp.

SELECT TOP 10 PERCENT [First Name], [Last Name] _ FROM Students _ WHERE [Graduation Year] = 1994 _ ORDER BY [Grade Point Average] DESC;

Mệnh đề WHERE

Mệnh đề WHERE xác định những bản ghi từ các bảng được liệt kê trong mệnh đề FROM

Microsoft Jet Database engine chọn các bản ghi thoả mãn điều kiện liệt kê trong mệnh đề WHERE. Nếu bạn không có mệnh đề WHERE, truy vấn sẽ trả lại tất cả các dòng từ các bảng được chọn. Nếu bạn chỉ ra hơn một bảng trong truy vấn và không có mệnh đề WHERE hoặc mệnh đề kết nối JOIN, truy vấn của bạn sẽ trả lại kết quả tích đề các của các bảng.

Chú ý: Mặc dù mệnh đề WHERE có thể đảm nhận tác vụ kết nối các bảng, nhưng bạn phải dùng một mệnh đề JOIN để thực hiện các thao tác kết nối nhiều bảng với nhau nếu bạn muốn kết quả của truy vấn có thể cập nhật được.

Mệnh đề WHERE tương tự như HAVING. WHERE xác định những bản ghi được chọn. Một cách tương tự, một khi các bản ghi đã được nhóm bởi GROUP BY, HAVING sẽ quyết định những bản ghi nào được hiển thị.

Dùng mệnh đề WHERE để loại bỏ các bản ghi bạn không muốn nhóm bởi GROUP BY.

Một mệnh đề WHERE có thể có tới 40 biểu thức được kết nối bởi các toán tử logic như AND, OR.

Khi bạn dùng một trường tên có chứa dấu cách hoặc dấu câu, bạn phải bỏ tên trường đó trong dấu ngoặc vuông ([]):

SELECT [Product ID], [Units In Stock] _ FROM Products _ WHERE [Units In Stock] <= [Reorder Level];

Khi bạn xác định đối số điều kiện, ngày phải định dạng theo dạng của Mỹ, ngay cả khi bạn không dùng phiên bản Jet database của Mỹ. Ví dụ, May 10, 1994, được viết là 10/5/94 theo kiểu Anh và 5/10/94 theo kiểu Mỹ. Để chắc chắn, cần đặt ngày của bạn vào trong cặp dấu thăng (#), như ví dụ sau đây:

Để tìm các bản ghi có ngày May 10, 1994 trong một CSDL Anh, bạn phải dùng câu lệnh sau đây:

SELECT * FROM Orders _ WHERE [Shipped Date] = #5/10/94#;

Bạn có thể dùng hàm DateValue, nó nhận biết được mọi định dạng được thiết lập bởi Microsoft Windows. Ví dụ, mã sau cho chuẩn ngày Mỹ:

SELECT * FROM Orders _ WHERE [Shipped Date] = DateValue('5/10/94');
Mã sau dùng cho Anh:
SELECT * FROM Orders _ WHERE [Shipped Date] = DateValue('10/5/94');
Mệnh đề GROUP BY:
GROUP BY là mệnh đề tuỳ chọn cho phép kết hợp các bản ghi theo một giá trị giống hệt nhau của một trường trường xác định vào trong một bản ghi duy nhất. Giá trị tổng hợp được tạo ra cho mỗi bản ghi nếu trong câu lệnh SQL có chứa hàm tổng hợp (aggregate function), như hàm Sum, Count.v.v.

Các bản ghi có giá trị rỗng ở trường nhóm vẫn được nhóm. Tuy nhiên, nó không được tổng hợp nếu trường tổng hợp là rỗng.

Dùng mệnh đề WHERE để loại bỏ các bản ghi bạn không muốn nhóm, và dùng mệnh đề HAVING để lọc các bản ghi sau khi chúng đã được nhóm.

Trừ khi dữ liệu là kiểu ký ức (Memo) hoặc trường tự động (Automation), một trường trong danh sách nhóm sau mệnh đề GROUP BY có thể tham chiếu tới bất kỳ trường nào được liệt kê trong mệnh đề FROM, thậm chí nếu trường đó không có trong mệnh đề SELECT. Jet database engine không thể nhóm trên các trường Memo hoặc Automation.

Tất cả các trường được liệt kê sau SELECT phải hoặc bao gồm trong danh sách trường nhóm hoặc ;à một hàm nhóm (aggregate function).

Mệnh đề HAVING

Xác định những bản ghi được nhóm nào được hiển thị trong mệnh đề SELECT với một mệnh đề GROUP BY. Một khi mệnh đề GROUP BY kết hợp các bản ghi, HAVING hiển thị bất cứ bản ghi nào được nhóm thoả mãn điều kiện trong mệnh đề HAVING.

HAVING tương tự WHERE, nó quyết định những bản ghi nào được chọn. Một khi các bản ghi được nhóm bởi GROUP BY, HAVING xác định bản ghi nào được hiển thị.

HAVING là một mệnh đề tuỳ chọn. Một mệnh đề HAVING có thể có tới 40 biểu thức được kết hợp các toán tử logic như AND và OR.

Mệnh đề ORDER BY

Mệnh đề ORDER BY xác định thứ tự sắp xếp của các bản ghi trong truy vấn. Trong mệnh đề ORDER BY, bạn xác định một trường hay các trường được dùng làm khoá sắp xếp, và sau đó xác định các bản ghi xuất hiện theo thứ tự độ lớn tăng dần hay giảm dần. Ví dụ sau trả lại tất cả các bản ghi trong bảng Employees được liệt kê họ theo thứ tự ABC :

SELECT * FROM Employees ORDER BY [Last Name] ASC;

Trong ví dụ này, ASC là tuỳ chọn - thứ tự sắp xếp mặc định là tăng dần. Tuy nhiên, bạn có thể thêm từ khoá ASC và cuối mỗi trường bạn muốn sắp theo thứ tự tăng dần.

Để sắp theo thứ tự giảm dần, thêm từ khoá DESC vào cuối các trường bạn muốn sắp theo thứ tự giảm dần.

Bạn cũng có thể dùng cũng có thể dùng số thứ tự của trường được chọn trong mệnh đề SELECT để chỉ ra trường làm khoá sắp xếp:

SELECT [First Name], [Last Name] FROM Employees ORDER BY 2 ASC;

Bạn cũng có thể sắp xếp theo nhiều trường. Các bản ghi trước hết được sắp theo trường đầu tiên trong danh sách các trường sắp xếp. Các bản ghi có cùng giá trị trên trường đó lại tiếp tục được sắp xếp trên trường tiếp theo trong danh sách và quá trình lại có thể lặp lại nếu tồn tại một số bản ghi có cùng giá trị tại trường vừa rồi.

Ví dụ sau chọn trường lương và sắp xếp giảm dần, tất cả các nhân viên cùng lương sẽ được sắp họ tăng dần theo thứ tự ABC.

SELECT [Last Name], Salary FROM Employees _ ORDER BY Salary DESC, [Last Name];

ORDER BY thường là thành phần cuối cùng trong câu lệnh SQL. Nó là mệnh đề tuỳ chọn (trừ khi bạn dùng chỉ dẫn TOP hoặc TOP n PERCENT trong mệnh đề SELECT).

Mệnh đề WITH OWNERACCESS

Trong môi trường đa người dùng với nhóm làm việc có chia quyền, dùng WITH OWNERACCESS cuối mỗi truy vấn để trao cho người dùng, người thi hành truy vấn cho phép xem dữ liệu trong truy vấn dầu rằng người đó mặt khác bị hạn chế xem các bảng cơ bản của CSDL.

Ví dụ sau cho phép người dùng trả lại thông tin về lương, thậm chí nếu người dùng không được phép xem bảng kết toán, là kết quả của một truy vấn do một người khác có đủ quyền thi hành.

SELECT [Last Name], [First Name], Salary FROM Employees _ ORDER BY [Last Name] _ WITH OWNERACCESS OPTION;

Nếu một người dùng mặt khác bị ngăn không được tạo hoặc thêm vào một bảng, bạn có thể dùng WITH OWNERACCESS OPTION để cho phép người dùng thi hành một câu lệnh tạo hoặc nối bảng.

Tuỳ chọn này yêu cầu bạn truy cập vào file hệ thống System.mda được kết hợp với CSDL. Nó thực sự hữu dụng chỉ trong môi trường đa người dùng có chia quyền.

2.Dùng một truy vấn tạo bảng.

Một sự biến đổi trong câu lệnh SELECT cho phép bạn tạo ra một bảng mới, thay thế cho một đối tượng Recordset. Để làm điều này, bạn thêm mệnh đề INTO. Ví dụ sau tạo ra một bảng mới New Employees bởi truy vấn bảng Employees:

SELECT * INTO [New Employees] FROM Employees;

Bạn có thể dùng truy vấn tạo bảng để lấy các bản ghi, tạo một bảng dự phòng, hoặc làm một bản sao để đưa sang một CSDL khác hoặc dùng làm cơ sở cho các báo cáo hiển thị dữ liệu trong mỗi định kỳ. Ví dụ, bạn có thể tạo ra bản báo cáo bán hang từng tháng bằng cách thi hành truy vấn tạo bảng này mỗi tháng.

Bạn có thể muốn xác định một khoá chính cho bảng mới tạo. Khi bạn tạo bảng đó, các trường trong bảng mới sẽ thừa kế kiểu dữ liệu va kích thước của mỗi trường trong bảng cơ sở, nhưng các đặc tính khác của các trường hoặc bảng không được chuyển sang.

3 Dùng truy vấn xoá.

Tạo một truy vấn xoá để loại bỏ các bản ghi từ một hoặc các bảng được liệt trong mệnh đề FROM thoả mãn điều kiện trong mệnh đề WHERE, như cú pháp dưới đây:

DELETE [table *] FROM tableexpression WHERE criteria

DELETE đặc biệt hữu dụng khi bạn muốn xoá một lúc nhiều bản ghi.

Trong một câu lệnh xoá trên nhiều bảng, bạn phải bao gồm đối table. Nếu bạn xác định xoá bản ghi trên nhiều bảng, không có bảng nào chứa khoá chính của một quan hệ 1 - n.

Nếu muốn xoá toàn bộ bản ghi trong một bản, Dùng truy vấn xoá chính bảng đó còn nhanh hơn là dùng truy vấn xoá. Bạn có thể dùng một phương thức thi hành với một câu lệnh DROP TABLE để xoá bảng đó ra khỏi CSDL. Tất nhiên nếu bạn xoá bảng thì mất luôn cả cấu trúc. Ngược lại, khi dùng truy vấn xoá thì chỉ phần dữ liệu bị xoá; Cấu trúc bảng và các thuộc tính của các trường vẫn còn nguyên vẹn.

Bạn có thể dùng DELETE để xoá bán ghi trong một báng hoặc báng liên kết bên n cúa một quan hệ 1 - n. Thao tác xoá theo tầng trong truy vấn chỉ xoá bảng bên n của quan hệ . Ví dụ, trong quan hệ giữa bảng Customers và bảng Orders, bảng Orders là phía n, nên thao tác xoá chỉ ảnh hưởng đến bảng Orders.

Một truy vấn xoá xoá toàn bộ bản ghi, không xoá chọn lọc theo các trường. Nếu bạn muốn xoá dữ liệu trong một trường cụ thể nào đó, dùng truy vấn cập nhật (UPDATE) để thiết lập dữ liệu trường đó là rỗng (NULL).

Một khi bạn đã loại bỏ các bản ghi bằng cách dùng truy vấn xoá, bạn không thể khôi phục lại thao tác của mình. Nếu bạn muốn biết những bản ghi đã được xoá, trước hết, kiểm tra kết quả của truy vấn chọn có cùng điều kiện (với truy vấn xoá) và sau đó tiến hành truy vấn xoá.

Bất cứ lúc nào bạn cũng nên sao lưu dữ liêu phòng khi bạn xoá nhầm các bản ghi.

4.Dùng truy vấn bổ sung.

Bạn có thể dùng mệnh đề INSERT INTO để thêm các bản ghi vào bảng hay tạo một truy vấn bổ sung.

Bạn có thể dùng những cú pháp sau đây để thực hiện truy vấn bổ sung nhiều bản ghi:

INSERT INTO target [IN externaldatabase] SELECT [source.]field1[, field2[, ...] FROM tableexpression

Ngược lại, dùng cú pháp sau để thực hiện truy vấn bổ sung một bản ghi:

INSERT INTO target [(field1[, field2[,...]])] VALUES (value1[, value2[, ...])

Bạn có thể dùng mệnh đề INSERT INTO để thêm một bản ghi đơn vào một bảng dùng cú pháp truy vấn bổ sung bản ghi đơn. Trong trường hợp này, Câu lệnh phải xác định tên và giá trị cho mỗi trường của bản ghi. Bạn phải xác định các trường của bản ghi mà dữ liệu se được gán vào cũng như giá trị của trường đó. Khi bạn không chỉ rõ danh sách trường, giá trị mặc định hoặc NULL sẽ điền vào các trường vắng mặt. Các bản ghi được thêm vào cuối bảng.

Bạn cũng có thể dùng INSERT INTO để nối một tập hợp các bản ghi từ một bảng khác hoặc một truy vấn dùng mệnh đề SELECT ... FROM được chỉ ra ở cú pháp trên. Trong

trường hợp này, mệnh đề SELECT chỉ rõ trường nối thêm vào bảng đích.

Bảng nguồn hoặc bảng đích có thể là một bảng hay một truy vấn. Nếu một truy vấn được xác định, Microsoft Jet database engine nối một tập hợp bản ghi vào một hoặc nhiều bảng được chỉ ra trong truy vấn.

INSERT INTO là một tuỳ chọn, nhưng khi có mặt nó, phải đứng trước SELECT.

Nếu bảng đích chứa khoá chính, phải chắc chắn rằng bạn bổ sung những bản ghi với nội dung khoá là duy nhất, và trường đó không được để trống.

Nếu bạn bổ sung các bản ghi vào một bảng với trường Couter, loại bỏ trường đó ra khỏi danh sách nếu bạn muốn Microsoft Jet đánh số lại các bản ghi. Thêm trường Counter vào truy vấn nếu bạn muốn giữ lại giá trị ban đầu. Nếu có giá trị trùng, tất nhiên là Jet database sẽ không bổ sung bản ghi đó.

Dùng mệnh đề IN để nối các bản ghi vào một bảng của một CSDL khác.

Để tạo một bảng mới, dùng mệnh đề SELECT ... INTO thay thế để tạo một truy vấn tạo bảng.

Để tìm ra các bản ghi sẽ được bổ sung, trươc khi thi hành truy vấn bổ sung, hãy xem qua kết quả của truy vấn chọn với cùng biểu thức điều kiện.

Một truy vấn bổ sung không ảnh hưởng tới các bảng hoặc truy vấn nguồn.

5.Truy vấn cập nhật:

Truy vấn UPDATE thay đổi giá trị trong các trường được thoả mãn các điều kiện cập nhật.

UPDATE table SET newvalue WHERE criteria;

UPDATE đặc biệt hữu dụng khi bạn muốn thay đổi nội dung nhiều bản ghi hoặc khi các bản ghi bạn muốn thay đổi nằm trên nhiều bảng. Thông thường bạn dùng truy vấn này với một phương thức thi hành.

Bạn có thể thay đổi một vài trường cùng lúc. Ví dụ sau tăng Order Amount lên 10% và giá trị Freight lên 3%:

UPDATE Orders _ SET [Order Amount] = [Order Amount] * 1.1, _ Freight = Freight * 1.03 _
WHERE [Ship Country] = 'UK';

UPDATE không tạo ra tập kết quả. Nếu bạn muốn xem những bản ghi nào được cập nhật, trước tiên xem kết quả của truy vấn chọn (dùng cùng biểu thức điều kiện) và sau đó thi hành truy vấn cập nhật.
Truy vấn Crosstab:
Truy vấn Crosstab cho phép bạn chọn các giá trị từ các trường hay các biểu thức như là các tiêu đề cột, vì thế, bạn có thể xem dữ liệu một cách cô đọng hơn với một câu lệnh SELECT bình thường. Bạn dùng mệnh đề TRANSFORM để tạo các truy vấn Crosstab.

TRANSFORM aggfunction selectstatement PIVOT pivotfield [IN (value1[,value2[,...]])]

Mệnh đề TRANSFORM dùng những tham số sau: aggfunction Một hàm tổng hợp trên dữ liệu được chọn.

selectstatement : Một câu lệnh SELECT
pivofield : Trường hoặc biểu thức bạn muốn dùng để tạo tiêu đề cột trong kết quả của truy vấn.
value1, value2: Các giá trị cố định được dùng để tạo tiêu đề cột.
Khi bạn tổng hợp dữ liệu với công cụ là truy vấn Crosstab, bạn chọn các giá trị các trường hay biểu thức cụ thể như là các tiêu đề vì thế bạn có thể xem dữ liệu trong một dạnh cô đọng hơn.

TRANSFORM đứng trước mệnh đề SELECT xác định các tiêu đề dòng và đứng trước một mệnh đề GROUP BY xác định các dòng cụ thể được nhóm. Tất nhiên, bạn có thể tuỳ chọn các mệnh đề khác, như WHERE, những mệnh đề xác định thêm các tiêu chuẩn chọn hay sắp xếp.

Các giá trị trả lại trong pivotfield được dùng như là tiêu đề cột trong tập kết quả của truy vấn. Ví dụ, việc xoay doanh số bán hàng trong một tháng trong một truy vấn Crosstab sẽ tạo ra 12 cột. Bạn có thể hạn chế pivotfield để tạo các tiêu đề cột từ các gía trị cố định (giá trị 1, giá trị 2) được liệt kê trong mệnh đề tuỳ chọn IN.

Ví dụ sau tạo ra một truy vấn Crosstab trình bày kết quả bán hàng theo tháng trong một năm của một người nào đó. Các tháng được trả lại là các cột từ trái sang phải, và tên sản phẩm được liệt kê từ trên xuống dưới như các hàng.

PARAMETERS [Sales for which year ?] LONG; TRANSFORM _ Sum([Order Details].Quanlity * ([Order Details].[Unit Price] _ -([Order Details].Discount / 100) * [Order Details].[Unit Price])) AS Sales _ SELECT [Product Name] FROM Orders INNER _ JOIN (Products INNER JOIN [Order Details] ON Products.[Product ID] = _ [Order Details]. [Product ID]) ON Orders.[Order ID] = _ [Order Details].[Order ID] WHERE DatePart _ ("yyyy", [Order Date]) = [Sales for which year ?] _ GROUP BY [Product Name] ORDER BY [Product Name] _ PIVOT DatePart("m", [Order Date]);

Kết nối:

Một trong những đặc tính hữu ích nhất của CSDL quan hệ là khả năng nối hai hay nhiêu bảng với nhau để tạo nên một bảng mới (hay một Recordset) chứa đựng thông tin từ các bảng cũ.

Các bảng được nối theo các mối quan hệ giữa chúng, thông thường nhất là giữa khoá chính của một bảng và khoá ngoài tương ứng của bảng kia. Tuỳ thuộc vào cách thức nối các bảng với nhau mà bạn có thể tạo ra các loại liên kết sau:

INNER JOIN Các bản ghi của cả hai bảng được chứa trong liên kết chỉ khi một trường cụ thể trong bảng khớp với một trường cụ thể trong bảng thứ hai.

LEFT OUTER JOIN Tất cả các bản ghi từ bảng một được chứa trong liên kết, cùng với các bản ghi trong bảng hai mà ở đó các trường cụ thể khớp với các trường tương ứng trong bảng một.

RIGHT OUTER JOIN Tất cả các bản ghi từ bảng thứ hai được đưa vào liên kết cùng với các bản ghi từ bảng một mà có các trường khớp với các trường trong bảng hai.

Inner joins:

Để tạo ra một truy vấn chỉ chứa các bản ghi có dữ liệu trong trường liên kết giống nhau, hãy dùng phép liên kết INNER JOIN.

INNER JOIN kết hợp các bản ghi của hai bảng khi có các giá trị khớp nhau trong trường

liên kết. Dùng cú pháp sau:

FROM table1 INNER JOIN table2 ON table1.field = table2.field2

Bạn có thể dùng INNER JOIN trong bất cứ mệnh đề FROM nào. Nó tạo ra một liên kết tương đương (Equi-joins) , như một liên kết INNER JOIN. Equi-joins là một dạng kết nối phổ biến. Chúng kết hợp các bản ghi từ hai bảng khi có dữ liệu trong hai trường kết nối khớp nhau.

Bạn có thể dùng INNER JOIN với các bảng Department và Employees để chọn tất cả các nhân viên trong mỗi phòng. Ngược lại, để chọn tất cả các phòng (thậm chí nếu một số phòng không có nhân viên nào), bạn có thể dùng LEFT JOIN hoặc RIGHT JOIN.

Bạn có thể liên kết bất cứ hai trường kiểu số nào, thậm chí nếu chúng khác kiểu. Ví dụ bạn có thể kết nối một trường số, với thuộc tính kích thước được thiết lập là số nguyên, và một trường Couter. Ví dụ sau cho chúng ta biết kết nối hai bảng Categories và Products trên trường CategoryID như thế nào:

SELECT [Category Name], [Product Name] _ FROM Categories INNER JOIN Products _ ON Categories.[Category ID] = Products.[Category ID];

Ví dụ sau tạo ra hai liên kết tương đương (equi-joins): một giữa hai bảng Order Details và Orders và một giữa hai bảng Orders và Employees. Điều này là cần thiết bởi bảng Employees không chứa thông tin về bán hàng, và Order Details không chứa thông tin về nhân viên. Truy vấn đưa ra một danh sách các nhân viên và tổng số về hàng họ bán được:

Dim MyQRY AS QueryDef MyQRY.SQL = "SELECT DISTINCTROW Sum([Unit Price]. [Quantity]) _ AS [Sales], [First Name] & " " & _ [Last Name] AS Name FROM Employees _ INNER JOIN (Order INNER JOIN [Order Details] _ ON Order.[Order ID] = [Order Details]. [Order ID]) _ ON Employees.[Employee ID] = Orders.[Employee ID] _ GROUP BY [First Name] & " " & [Last Name];"

Câu lệnh này phải được sử dụng như là thuộc tính của đối tượng QueryDef hoặc là tham số của phương thức OpenRecordset.

Ví dụ trên dùng chỉ dẫn DISTINCTROW để loại bỏ dữ liệu trên cơ sở các bản ghi hoàn toàn trùng nhau. Ví dụ, bạn có thể tạo một truy vấn liên kết các bảng Customers với bảng Orders với trường liên kết là trường Customer ID. Bảng Customers không chứa các bản ghi có cùng nội dung trường Customer ID, nhưng trong bảng Orders thì có bởi một khách hàng có thể có nhiều hợp đồng đặt mua hàng. Câu lệnh SQL sau chỉ ra cách bạn dùng chỉ dẫn DISTINCTROW để đưa ra một danh sách các công ty có ít nhất một hợp đồng đặt mua hàng nhưng không cụ thể về các hợp đồng đó.

SELECT DISTINCTROW [Company Name] _ FROM Customers INNER JOIN Order _ ON Customers.[Customer ID = Orders.[Customer ID] _ ORDER BY [Company Name];

Nếu bạn bỏ sót chỉ dẫn DISTINCTROW thì kết quả sẽ trả lại các tên công ty trùng nhau vì có nhiều bản hợp đồng đặt mua hàng.

DISTINCTROW chỉ có hiệu quả khi bạn chọn các trường từ một số bảng được liên kết trong truy vấn. DISTINCTROW được bỏ qua khi bạn chọn từ một bảng.

Liên kết trái, phải (LEFT JOIN & RIGHT JOIN - Liên kết ngoài):

Liên kết ngoài (OUTER JOIN) nối các bản ghi của bảng nguồn khi chúng được dùng

trong bất kỳ mệnh đề FROM nào với cú pháp sau:

FROM table1 [LEFT | RIGHT] JOIN table2 ON table1.field1 = table2.field2

Dùng LEFT JOIN để tạo một liên kết ngoài bên trái. Liên kết ngoài bên trái gồm tất cả các bản ghi từ bảng thứ nhất (bảng bên trái), tức là bao gồm cả các bản ghi không khớp nội dung trường liên kết với bất cứ bản ghi nào của bảng thứ hai (bảng bên phải).

Dùng RIGHT JOIN để tạo liên kết ngoài bên phải. Liên kết ngoài bên phải bao gồm tất cả các bản ghi có trong bảng thứ hai (bảng bên phải), tức là bao gồm cả các bản ghi không khớp nội dung trương liên kết với bất cứ bản ghi nào của bảng còn lại (bảng bên trái).

Ví dụ, bạn có thể dùng LEFT JOIN cho bảng Departments (trái) và bảng Employees (phải) để lấy ra tất cả các phòng ban, trong đó có cả các phòng không có nhân viên nào. Để lấy tất cả các nhân viên, bao gồm cả những nhân viên không thuộc phòng ban nào, hãy dùng RIGHT JOIN.

Ví dụ sau có thấy cách thức để bạn có thể liên kết báng Categories và báng Products dự trên trường Category ID. Truy vấn sẽ đưa ra danh sách tất cả tên các hạng mục, trong đó có nhưng hạng mục không có sản phẩm nào.

SELECT [Category Name], [Product Name] _ FROM Categories LEFT JOIN Products _ ON Categories.[Category ID] = Products.[Category ID];

Trong ví dụ trên đây, Category ID là một trường liên kết, nhưng không có mặt trong kết quả của truy vấn vì nó không được chọn trong mệnh đề SELECT.

Liên kết lồng nhau (Nested joins):

Bạn có thể lồng các chỉ dẫn JOIN như cú pháp sau đây:

SELECT fields FROM table1 INNER JOIN (table2 INNER JOIN [(]table3 _ [INNER JOIN [(] tablex [INNER JOIN ...)] _ ON table3.field3 = tablex.fieldx)] _ ON table2.field2 = table3.field3) _ ON table1.field1 = table2.field2;

Truy vấn hợp (Union Queries): Bạn có thể dùng các thao tác nối để tạo các truy vấn hợp, kết hợp kết quả của hai hay nhiều bảng hoặc truy vấn độc lập. query1 UNION [ALL] query2 [UNION [ALL] queryn [...]]

query đại diện cho biểu thức xác định trường chứa dữ liệu kiểu số mà bạn muốn tính trung bình hoặc một biểu thức thực hiện một phép tính có dùng dữ liệu trong trường đó. Toán hạng trong biểu thức có thể tên của các trường, có thể là một hằng, một hàm (có thể là có sẵn của hệ thống hoặc do người dùng định nghĩa nhưng không được là hàm tổng hợp(aggregate functions)).

Bạn có thể kết hợp kết quả của một truy vấn và một câu lệnh SQL trong một phép hợp UNION đơn. Ví dụ sau kết hợp kết quả của một truy vấn có tên gọi New Accounts và một câu lệnh SELECT: TABLE [New Accounts] UNION ALL _ SELECT * FROM Customers WHERE [Order Amount] > 1000;

Mặc định, không có hai bản ghi nào trùng nhau trong kết quả của một truy vấn hợp; tuy nhiên, bạn có thể bao gồm cả chỉ dẫn ALL để kết quả có cả các bản ghi trùng nhau. Nó cũng làm cho truy vấn thi hành nhanh hơn.

Tất cả các truy vấn trong phep hợp UNION yêu cầu có cùng số trường, tuy nhiên các trường không cần thiết phải có cùng kích thước hoặc kiểu dữ liệu.

Bạn có thể dùng mệnh đề GROUP BY và/hoặc HAVING trong mỗi truy vấn tham gia phép hợp để nhóm dữ liệu. Bạn có thể dùng một mệnh đề ORDER BY ở cuối truy vấn cuối cùng để kết quả trả về được sắp theo một thứ tự xác định.

7.Tối ưu truy vấn:
Tối ưu CSDL là một chủ đề lớn và bao hàm nhiều vấn đề. Nhiều yếu tố, bao gồm cả cấu hình phần cứng và phần mềm, cài đặt Windows, bộ đệm, không liên quan đến các truy vấn, nhưng chúng ảnh hưởng đến hiệu quả của CSDL.
Một yếu tố quan trọng khác là dùng các bảng gắn với CSDL hơn là kết nối trực tiếp với CSDL từ xa. Vấn đề này được bàn kỹ ở trong cuốn "Accessing External Data".
Phần này sẽ trình bày ngắn gọn một số cách tối ưu liên quan đặc biệt tới sự xây dựng các truy vấn, và tận dụng khả năng tối ưu Jet database để xây dựng các truy vấn.
Một số chiến lược tối ưu thường dùng:

- Hạn chế dùng các trường sắp xếp, đặc biệt là các trường không được tạo chỉ mục.
- Đảm bảo các bảng kết nối từ các nguồn khác nhau đều được tạo chỉ mục hoặc là được liên kết trên các khoá chính.
- Nếu dữ liệu không thay đổi thường xuyên, dùng các truy vấn tạo bảng từ truy vấn Dynasets của bạn. Dùng các bảng để làm cơ sở cho các form, các báo cáo hơn là dùng các truy vấn.
- Nếu bạn đang tạo một truy vấn Crosstab, hãy dùng tiêu đề cột cố định bất cứ khi nào có thể.
- Dùng BETWEEN...AND, IN và các dấu "=" trên các cột được đánh chỉ mục.
- Khi tạo một truy vấn, không nên thêm các trường không cần thiết.

Tối ưu các truy vấn với công nghệ Rushmore:
Rushmore là một công nghệ truy cập dữ liệu được dùng trong Jet database engine cho phép một tập hợp các bản ghi được truy cập một cách hết sức hiệu quả. Với Rushmore, khi bạn dùng một kiểu biểu thức nhất định trong điều kiện truy vấn, truy vấn của bạn sẽ thi hành nhanh hơn rất nhiều.
Biểu thức tối ưu ở dạng đơn giản:

Jet database engine có thể tối ưu hoá các biểu thức đơn giản trong mệnh đề WHERE trong câu lệnh SELECT. Dạng biểu thức đơn giản có thể thành lập nên toàn bộ hoặc một phần phần của biểu thức.
Một biểu thức đơn giản có thể tối ưu hoá có một trong những dạng sau:

indexedfield comparisonoperator expression - or -expression comparisonoperator indexedfield Trong dạng biểu thức đơn giản nhất có thể tối ưu:

- indexedfield phải là một trường mà trên đó có tạo lập chỉ mục.
- comparisonoperator phải là một trong những kiểu sau đây: <, >, =, >=, <=, <>, BETWEEN, LIKE, hoặc IN.
- expression là một biểu thức hợp lệ bất kỳ, bao gồm các hằng, các hàm, các trường từ các bảng.

Chú ý: Để kết quả là tốt nhất, giá trị so sánh trong biểu thức dùng LIKE phải bắt đầu bằng một ký tự chứ không được là ký tự đại diện (*, ?). Bạn có thể tối ưu LIKE "m*" chứ không thể tối ưu LIKE "*m*".

Biểu thức có thể tối ưu ở dạng phức hợp.
Microsoft Jet dùng Rushmore để tối ưu hoá các biểu thức phức hợp được kết hợp từ các biểu thức (có thể tối ưu) đơn giản với các toán tử AND hoặc OR. Một biểu thức như thế nằm trong các dạng sau:

- simpleexpression AND simpleexpression
- simpleexpression OR simpleexpression

Ghi nhớ những điều sau đây khi dùng biểu thức tối ưu hoá Rushmore:

- Hàm COUNT(*) được tối ưu ở mức cao cho các truy vấn dùng Rushmore.
- Nếu chỉ mục là giảm dần và toán tử so sánh khác dấu bằng, truy vấn không thể tối ưu được.
- Rushmore sẽ làm việc với Microsoft Access tables, cũng như với Microsoft FoxPro