NGÔN NGỮ TRUY VẤN SQL

NỘI DUNG

- Giới thiệu
- Tạo truy vấn
- Cú pháp của các lệnh truy vấn

Giới thiệu

Ngôn ngữ truy vấn có cấu trúc (SQL _ *Structure Query Language*) là một ngôn ngữ thường dùng để truy vấn, cập nhật và quản lý các cơ sở dữ liệu quan hệ *(relational database)* như là *Access.* SQL gồm các lệnh, mệnh đề, toán tử và các hàm aggregate (hàm tính gộp theo nhóm), các phần tử này kết hợp lại để thành các phát biểu SQL.

SQL được chia làm hai thành phần: DDL (Data Definition Language _ *Ngôn ngữ định nghĩa dữ liệu*) và DML (Data Manipulation Language _ *Ngôn ngữ thao tác dữ liệu*).

Các lệnh của DDL cho phép bạn tạo và định nghĩa các cơ sở dữ liệu *(database),* các thuộc tính *(field)* và các chỉ mục *(index)*.

Còn các lệnh của DML cho phép bạn xây dựng các truy vấn *(query)* để thao tác với dữ liệu.

Giới thiệu

- Các phát biểu của DDL là các biểu thức xây dựng từ các lệnh SQL sau:

CREATE	Dùng tạo mới các bảng, các thuộc tính và các chỉ mục
DROP	Dùng xóa các bảng và các chỉ mục từ cơ sở dữ liệu
ALTER	Dùng hiệu chỉnh các bảng bằng cách thêm vào các thuộc tính hay thay đổi định nghĩa của các thuộc tính.

- Các phát biểu của DML là các biểu thức xây dựng từ các lệnh SQL sau:

SELECT	Dùng truy vấn các mẫu tin đã thỏa điều kiện chỉ định.
INSERT	Dùng chèn một loạt dữ liệu vào cơ sở dữ liệu bằng một thao tác.
UPDATE	Dùng thay đổi các giá trị của các mẫu tin hay thuộc tính nào đó.
DELETE	Dùng xóa các mẫu tin từ một bảng.

Giới thiệu

- Các mệnh đề dùng hiệu chỉnh các điều kiện dùng định nghĩa dữ liệu muốn chọn hay thao tác. Gồm các mệnh đề sau:

FROM	Dùng chỉ định tên bảng, nơi các mẫu tin được chọn.
WHERE	Dùng chỉ định điều kiện các mẫu tin phải thỏa để được chọn.
GROUP BY	Dùng phân chia các mẫu tin thành các nhóm riêng biệt.
ORDER BY	Dùng sắp xếp các mẫu tin được chọn theo một thứ tự ấn định.

Tạo truy vấn thao tác dữ liệu

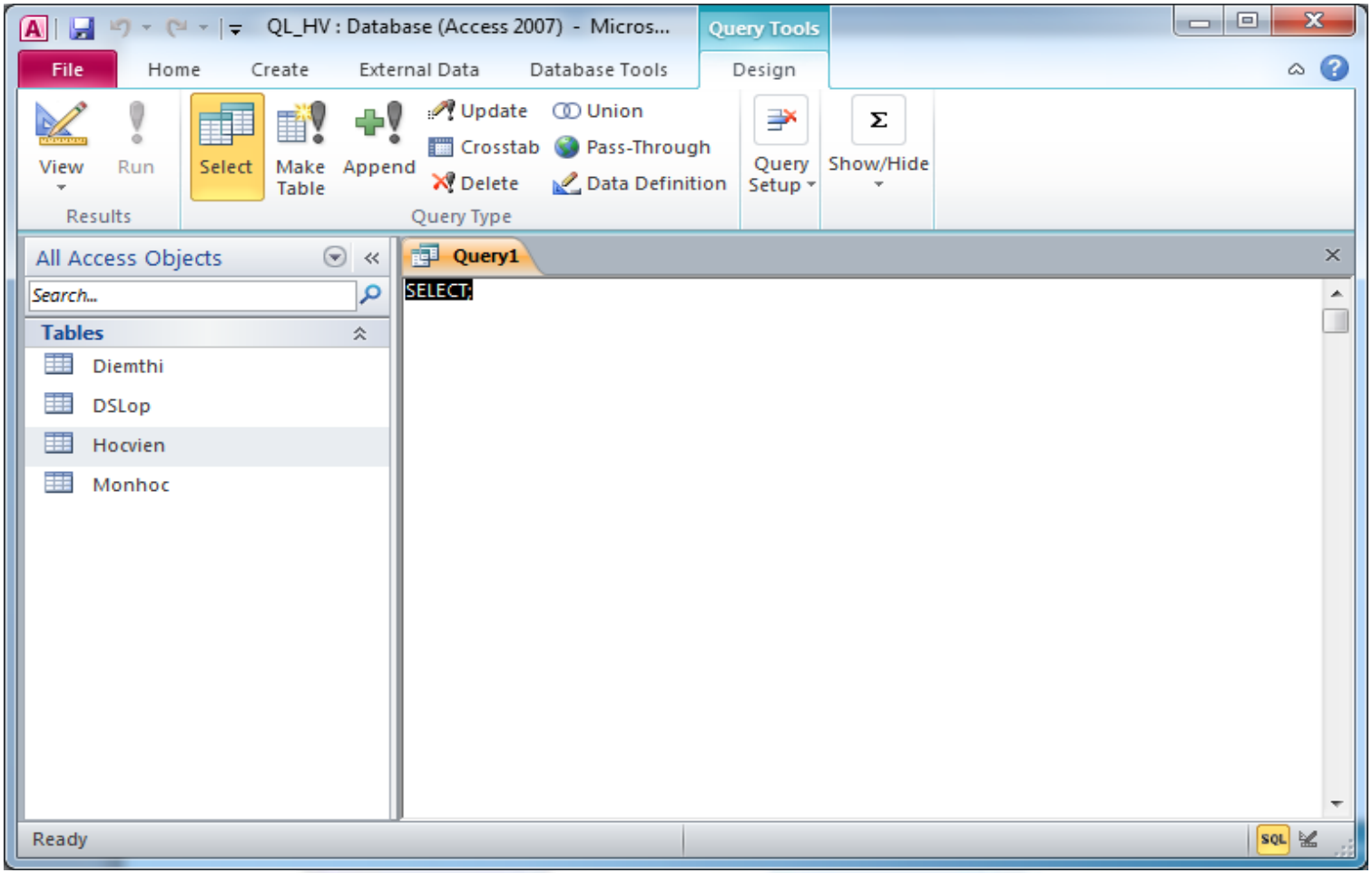
- Cách mở cửa sổ thiết kế truy vấn SQL

Bước 1: Chọn Create → Query Design. Sau bước này cửa sổ Show tables xuất hiện (như trong phần thiết kế truy vấn QBE).

Bước 2: Đóng cửa sổ Show table này lại, chọn SQL View . Cửa sổ thiết kế SQL xuất hiện, và chúng ta soạn câu lệnh SQL trong cửa sổ này.

Bước 3: Để thực hiện câu lệnh SQL chọn biểu tượng ! trên thanh công cụ.

Màn hình tạo câu truy vấn SQL



Cú pháp của câu lệnh SELECT

SELECT [DISTINCT|ALL]{*| <danh sách các thuộc tính>|<biểu thức cột> [AS <tên mới>]}
FROM <tên bảng>[<bí danh>] [, …]
[WHERE <điều kiện lựa chọn các bản ghi>]
[GROUP BY <danh sách tên cột dùng để gộp nhóm>] [HAVING <điều kiện lựa chọn nhóm>]
[ORDER BY <danh sách cột dùng để sắp xếp>]

Trong đó:

- biểu thức cột: là tên của một cột hoặc một biểu thức.
- tên bảng: là tên của một bảng hoặc một khung nhìn.
- bí danh: là một tên viết tắt của tên bảng.
- GROUP BY dùng để nhóm các bản ghi có cùng giá trị trong danh sách tên cột dùng để gộp nhóm thành một nhóm.
- HAVING dùng để lọc các nhóm thỏa mãn điều kiện lựa chọn nhóm.
- ORDER BY qui định thứ tự các bản ghi trong kết quả trả ra, thứ tự các bản ghi được sắp xếp dựa vào giá trị trong của các cột trong danh sách cột dùng để sắp xếp.

* Khi viết lệnh SQL cần chú ý:

- Phần đặt trong cặp dấu [ ]: là phần có thể sử dụng hoặc không.
- Những phần đặt trong cặp dấu {} : bắt buộc phải lựa chọn một số giá trị trong cặp dấu ngoặc này.
- Phần đặt trong cặp dấu <>: là phần bắt buộc phải có khi viết lệnh SQL.

Một số loại điều kiện có thể đặt sau từ khóa WHERE

- So sánh giá trị của hai biểu thức.
- Kiểm tra xem giá trị của một biểu thức có nằm trong một khoảng giá trị đã xác định không.
- Kiểm tra xem giá trị của một biểu thức có bằng một trong các giá trị thuộc một tập hợp đã xác định không.
- Kiểm tra xem một cột có chứa giá trị null không.

Một số toán tử dùng trong SQL

Các toán tử logic dùng để nối các biểu thức, thường dùng trong mệnh đề **WHERE**, gồm các toán tử **AND**, **OR** và **NOT**.

Các toán tử so sánh dùng để so sánh giá trị quan hệ của hai biểu thức, gồm các toán tử sau:

Toán tử	Ý nghĩa
<, <=, >, >=, =, <>	nhỏ hơn, nhỏ hơn hay bằng, lớn hớn, lớn hơn hay bằng, khác (không bằng).
BETWEEN … AND…	(trong khoảng) dùng chỉ định một vùng giá trị
LIKE, IS NULL, IS NOT NULL	giống như, có giá trị null, có giá trị khác rỗng.
IN , NOT IN	thuộc vào, không thuộc vào.

- Kí hiệu % dùng để sánh hợp với mọi xâu. Kí hiệu _ (dấu nối dưới) dùng để sánh hợp với mọi kí tự.

Ví dụ 1 Cho biết mã nhân viên, họ tên, đơn vị và lương của những nhân viên có lương lớn hơn 2300000.

```
SELECT manv, hoten, ma_dv, luong
FROM nhan_vien
WHERE luong>2300000
```

Ví dụ 2 Tìm tên các nhân viên có mã đơn vị là ‘P4’ và lương thấp hơn 2500000.

```
SELECT manv, ho_ten, ma_dv, luong
FROM nhan_vien
WHERE ma_dv=’P4’ AND luong < 2500000
```

- Kết nối các bảng
  Trong trường hợp câu hỏi liên quan đến nhiều bảng chúng ta cần kết nối các bảng với nhau. Có 2 kiểu kết nối là kết nối trong (INNER JOIN) và kết nối ngoài (LEFT JOIN, RIGHT JOIN).

  Ví dụ, kết nối bảng nhanvien va bảng phong dựa trên điều kiện kết nối nhanvien.maphong=phong.maphong

  - Kết nối trong: một bộ trong bảng nhanvien sẽ được kết nối với một bộ trong bảng phong khi và chỉ khi maphong của hai bộ này bằng nhau.
  - Kết nối ngoài: Nếu nhanvien LEFT JOIN phong ON nhanvien.maphong=phong.maphong thì với một bộ t trong bảng nhanvien được ghép với một bộ k trong bảng phong nếu hai bộ này có maphong bằng nhau, nếu không tìm thấy bộ nào trong bảng phong có maphong bằng maphong của bộ t thì bộ t được ghép với một bộ có giá trị null trong bảng phong. Kết nối nhanvien RIGHT JOIN phong ON nhanvien.maphong = phong.maphong cho kết quả bằng kết nối phong LEFT JOIN nhanvien ON nhanvien.maphong=phong.maphong.

- Ví dụ 1: Tìm những ngoại ngữ mà chưa có nhân viên nào học?

  Cách 1: SELECT mann, tenn
  FROM dmnn LEFT JOIN tdnn ON dmnn.mann=tdnn.mann
  WHERE tdnn.mann IS null

  Cách 2: SELECT mann, tenn
  FROM dmn
  WHERE mann NOT IN (SELECT DISTINCT mann FROM tdnn)

- Ví dụ 2: Cho biết mã nhân viên, họ tên, lương của các nhân viên làm ở phòng "kỹ thuật"?

  Cách 1: SELECT manv, hoten, luong
  FROM nhanvien INNER JOIN phong ON nhanvien.maphong=phong.maphong
  WHERE tenphong="kỹ thuật"

  Cách 2:SELECT manv, hoten, luong
  FROM nhanvien, phong
  WHERE nhanvien.maphong = phong.maphong AND tenphong="kỹ thuật"

  Ví dụ 3:Cho biết mã nhân viên, họ tên, ngày sinh của những nhân viên có tiếng Anh đạt trình độ từ C trở lên.
  SELECT nhanvien.manv, hoten, ngaysinh
  FROM (nhanvien INNER JOIN tdnn ON nhanvien.manv=tdnn.manv)
  INNER JOIN dmnn ON tdnn.mann=dmnn.mann
  WHERE tennn='Anh' AND tdo>='C'

  Chú ý: Khi sử dụng nhiều bảng trong câu lệnh SQL với những thuộc tính xuất hiện ở nhiều bảng, khi sử dụng cần chỉ rõ chúng ta dùng thuộc tính đó ở bảng nào, bằng cách viết: <tên bảng>.<tên thuộc tính>.

- Sử dụng các hàm gộp, GROUP BY, HAVING

SQL sử dụng một số hàm gộp sau:

SUM(thuộc tính): Tính tổng giá trị của thuộc tính.

MAX(thuộc tính): Tính giá trị lớn nhất của thuộc tính.

MIN(thuộc tính): Tính giá trị nhỏ nhất của thuộc tính.

AVG(thuộc tính): Tính giá trị trung bình của thuộc tính.

COUNT(* | thuộc tính | **distinct** thuộc tính): Đếm số các bộ trong quan hệ.

COUNT(*): Đếm tất cả các bộ.

COUNT(thuộc tính): Đếm các bộ mà giá trị của thuộc tính khác null.

COUNT(**distinct** thuộc tính): Đếm các bộ mà giá trị của thuộc tính khác null và không trùng nhau.

Ví dụ 1: Cho biết số nhân viên, và tổng lương của toàn cơ quan?

```
SELECT COUNT(manv) AS sonv, SUM(luong) AS tongluong
FROM nhanvien
```

Ví dụ 2: Cho biết lương cao nhất, lương trung bình của mỗi đơn vị? Thông tin đưa ra gồm: mã phòng, tên phòng, lương cao nhất, lương trung bình của phòng đó.

```
SELECT phong.maphong, tenphong, MAX(luong) AS maxluong,
AVG(luong) AS luongtb
FROM nhanvien INNER JOIN phong ON
nhanvien.maphong=phong.maphong
GROUP BY phong.maphong, tenphong
```

Cú pháp câu lệnh INSERT

- Thêm mới một mẫu tin

  Cú pháp: INSERT INTO <tên bảng> **VALUES** (<giá trị>, …)

  Lệnh này sẽ thêm một bộ mới vào bảng <tên bảng>. Các giá trị: <giá trị>, … lần lượt được gán cho các thuộc tính theo thứ tự khai báo trong lệnh **CREATE TABLE**.

  Ví dụ: Thêm một lớp mới vào bảng Lop

  INSERT INTO Lop **VALUES** ('K4TA', 'Khóa 4 Tối A', 25)

  Chú ý: Trong trường hợp không xác định được đầy đủ tất cả các giá trị của thuộc tính, chúng ta phải mô tả một cách cụ thể danh sách các thuộc tính sẽ nhận giá trị theo cú pháp sau:

  INSERT INTO <tên bảng> (<thuộc tính>, …) **VALUES** (<giá trị>, …)

  Các thuộc tính không được liệt kê trong danh sách sẽ nhận giá trị null (nếu các thuộc tính này không bị ràng buộc giá trị NOT NULL).

  Ví dụ: Thêm một học viên mới vào bảng Lop

  INSERT INTO Hocvien (MaHV, Holot, Ten, Phai, Ngaysinh, Diachi)
  VALUES ('567', 'Lê Thị', 'Giao', False, 15/5/75, '48 Nguyễn Trãi, Q. I')

- Thêm mới nhiều mẫu tin

Cú pháp: INSERT INTO <tên bảng> SELECT <danh sách các cột> FROM <tên bảng chứa dữ liệu>

Ví dụ 1: Nối tất cả các mẫu tin của bảng New Customers vào bảng Customers.

INSERT INTO Customers **SELECT** [New Customers].* **FROM** New Customers

Ví dụ 2: Nối tất cả các nhân viên tập sự vào bảng Employees có thời gian tập sự trên 30 ngày

INSERT INTO Employees **SELECT** Trainees.*

FROM Trainees **WHERE** HireDate < Now() - 30;

Cú pháp câu lệnh UPDATE

Cú pháp: **UPDATE <tên bảng> SET <thuộc tính> = <giá trị mới>, … WHERE <điều kiện>**

Những giá trị mới được gán cho các thuộc tính phải thuộc miền giá trị tương ứng của các thuộc tính.

Lệnh **UPDATE** chỉ cập nhật dữ liệu cho các bộ nào thỏa mãn điều kiện theo sau từ khóa **WHERE**.

Ví dụ 1: Sửa ngày sinh của học viên có mã số 123 thành 25/10/79

UPDATE Hocvien **SET** Ngaysinh = 25/10/79
WHERE MaHV = '123'

Ví dụ 2: Tăng giá bán (Unit Price) cho tất cả các mặt hàng không còn sản xuất từ nhà cung cấp có mã số (Supplier ID) là 8 lên 10%

UPDATE Products **SET** UnitPrice = UnitPrice * 1.1
WHERE SupplierID = 8 **AND** Discontinued = No

Cú pháp câu lệnh DELETE

Cú pháp: **DELETE FROM** <tên bảng> **WHERE** <điều kiện>

Lệnh này loại bỏ ra khỏi bảng <tên bảng> tất cả các bộ thỏa mãn điều kiện theo sau từ khóa **WHERE**.

Ví dụ 1: Hủy bỏ các học viên có mã số không hợp lệ (có giá trị là null) trong bảng Hocvien.

DELETE FROM Hocvien **WHERE** MaHV is Null

Ví dụ 2: Hủy bỏ môn học Cơ sở dữ liệu (có MaMH = 'CSDL') trong bảng Monhoc.

DELETE FROM Monhoc **WHERE** MaMH = 'CSDL'

Chú ý:

- Các thao tác cập nhật dữ liệu có thể làm cho cơ sở dữ liệu không còn tính kết nối (cohenrence) vì đã vi phạm các ràng buộc được áp đặt lên cơ sở dữ liệu. Chẳng hạn:
- Khi thêm một bộ mới vào bảng Hocvien như trong ví dụ trên sẽ gây ra một lỗi sai, nếu đã có một học viên khác mang mã số '567' (vi phạm ràng buộc duy nhất).
- Khi xóa môn học Cơ sở dữ liệu trong bảng Monhoc, sẽ làm các bộ có MaMH = 'CSDL' trong bảng Ketqua không còn kết nối với bộ nào của bảng Monhoc và trở nên "mồ côi".
- Các hệ quản trị cơ sở dữ liệu có thể giải quyết tình trạng không kết nối trên theo một trong các cách sau:

  Không cho phép người sử dụng thực hiện thao tác hủy không hợp lệ đó (báo lỗi sai).

  Tự động loại bỏ các bộ "mồ côi" ra khỏi bảng Ketqua.
- Tóm lại, việc cập nhật dữ liệu của một bảng (thêm mới, sửa đổi nội dung, hủy bỏ) cần phải thỏa mãn tất cả các ràng buộc được định nghĩa trên bảng đó, nhằm đảm bảo tính kết nối của cơ sở dữ liệu.

Tạo các truy vấn định nghĩa dữ liệu (DDL)

Mỗi truy vấn định nghĩa dữ liệu chỉ gồm một phát biểu định nghĩa dữ liệu. Access sử dụng các phát biểu sau:

Lệnh	Ý nghĩa
CREATE TABLE	Tạo một bảng mới.
ALTER TABLE	Hiệu chỉnh bảng đã có.
DROP TABLE	Xóa bảng ra khỏi CSDL.
CREATE INDEX	Tạo chỉ mục cho một thuộc tính
DROP INDEX	Xóa chỉ mục từ một thuộc tính

Cách tạo truy vấn định nghĩa dữ liệu

Bước 1: Tạo một truy vấn mới nhưng không đưa các bảng vào truy vấn (bấm nút *Close* khi hộp thoại *Add Tables* xuất hiện, để đóng hộp thoại này lại).

Bước 2: Chọn menu *Query, SQL Specific,Data-Definition.* Cửa sổ *Data-Definition Query* xuất hiện.

Bước 3: Nhập các phát biểu SQL cho truy vấn định nghĩa dữ liệu.

Lưu ý: Không được chuyển một truy vấn định nghĩa dữ liệu sang một truy vấn nào khác, nếu không, các phát biểu SQL vừa nhập sẽ bị mất.

Bước 4: Bấm nút **!** (*Run)* để thực thi truy vấn. Access sẽ tạo một đối tượng mới.

3.2 Tạo cấu trúc cho bảng (câu lệnh CREATE TABLE)

Cú pháp:

CREATE TABLE <tên bảng>(<thuộc tính> <kiểu> [(<kích thước>)], …

CONSTRAINT <tên chỉ mục> **PRIMARY KEY** (<khóa chính>) [**UNIQUE** (<khóa>), …] [**FOREIGN KEY** (<khóa ngoại>) **REFERENCES** <bảng ngoại>, …])

=> Lệnh này sẽ tạo ra một bảng mới rỗng có cấu trúc gồm các thuộc tính: <thuộc tính> với các kiểu dữ liệu: <kiểu dữ liệu>

Trong đó:

Kiểu gồm:

+ Kiểu số nguyên: **Integer, Long**
+ Kiểu số thực**: Single**, **Double**
+ Kiểu chuỗi: **Text(n)** (chuỗi gồm n ký tự)
+ Kiểu ngày/giờ: **Date**,**Time**
+ Kiểu tiền tệ: **Currency**
+ Kiểu luận lý (true, false): **Yes/No**

Mệnh đề CONSTRAINT

+ PRIMARY KEY: khai báo khóa chính của bảng.

+ UNIQUE: khai báo các khóa chỉ định khác (nếu có).

+ FOREIGN KEY: khai báo khóa ngoại của bảng.

Ví dụ 1: Tạo bảng NHANVIEN gồm các thuộc tính:

MANV (Primary, AutoNumber),
NGAYSINH (Date/Time)
TEN (Text, Size: 12) HOLOT (Text, Size: 30)
DIACHI(Text, Size: 50) DIENTHOAI (Text, Size: 12)

Bước 1: Tạo truy vấn mới.

Bước 2: Chọn menu *Query, SQL Specific, Data Definition*. Trong cửa sổ *Data Definition Query,* nhập các phát biểu sau:

CREATE TABLE NHANVIEN(MANV COUNTER, TEN TEXT(12), HOLOT TEXT(30), NGAYSINH DATETIME, DIACHI TEXT(50), DIENTHOAI TEXT(12) **CONSTRAINT** MANV_P **PRIMARY KEY** (MANV));

Bước 3: Bấm nút *Run* để thực thi truy vấn định nghĩa dữ liệu. Access sẽ tạo bảng NHANVIEN và bạn sẽ thấy trong cửa sổ cơ sở dữ liệu tên bảng NHANVIEN này.

* Nếu truy vấn định nghĩa dữ liệu được lưu lại với tên *Data Definition Query* bạn sẽ thấy trong cửa sổ cơ sở dữ liệu tên truy vấn định nghĩa dữ liệu cùng với biểu tượng của truy vấn định nghĩa dữ liệu Data Definition Query.

Ví dụ 2: Tạo bảng (quan hệ) Hocvien gồm các thuộc tính mã học viên (MaHV), họ (Holot), tên (Ten), ngày sinh (Ngaysinh) có khóa chính *(PrimaryKey)* là MaHV.

CREATE TABLE Hocvien(MaHV Text(8), Ho Text(30), Ten Text(10), Ngaysinh DateTime),

CONSTRAINT PrimaryKey **PRIMARY KEY** (MaHV))

Ví dụ 3: Tạo bảng Lop gồm các thuộc tính mã lớp (MaLop), tên lớp (TenLop), sĩ số (Siso)

CREATE TABLE Lop(MaLop Text(8), TenLop Text(30), Siso Byte, **CONTSTRAINT** Khoachinh **PRIMARY KEY** (MaLop) **UNIQUE** (TenLop))

Hiệu chỉnh cấu trúc bảng (Câu lệnh ALTER TABLE)

Cú pháp: **ALTER TABLE** <tên bảng> **ADD** {[**COLUMN**] <thuộc tính> <kiểu> [(<kích thước>)] | [**CONSTRAINT** <chỉ mục>] } |

DROP {[**COLUMN**] <thuộc tính> | **CONSTRAINT** <tên chỉ mục>}

Với mỗi phát biểu, bạn chỉ có thể thêm hay xóa một thuộc tính.

ADD COLUMN: thêm một thuộc tính.

ADD CONSTRAINT: thêm chỉ mục trên nhiều trường (xem lại phần trên).

DROP COLUMN: xóa một thuộc tính.

DROP CONSTRAINT: xóa chỉ mục trên nhiều thuộc tính.

Ví dụ 1: Thêm thuộc tính Phai vào bảng Hocvien

ALTER TABLE Hocvien **ADD** Phai YesNo;

Ví dụ 2: Xóa thuộc tính NGAYSINH trong bảng Hocvien

ALTER TABLE Hocvien **DROP COLUMN** NgaySinh;

Xóa bảng (Câu lệnh DROP TABLE)

Cú pháp:

DROP TABLE <tên bảng>

Ví dụ: Xóa bảng Hocvien:

DROP TABLE Hocvien

Tạo chỉ mục (Câu lệnh CREATE INDEX)

Cú pháp: **CREATE [UNIQUE] INDEX** <tên chỉ mục> **ON** <bảng> (<thuộc tính>), … [**WITH** {**PRIMARY** | **DISALLOW NULL** | **IGNORE NULL** }]

Trong đó:

UNIQUE: các giá trị của thuộc tính làm chỉ mục không được trùng lặp nhau.

PRIMARY: tạo chỉ mục là khóa chính, khi chọn **PRIMARY** có thể bỏ qua **UNIQUE**.

DISALLOW NULL: thuộc tính làm chỉ mục không được nhận giá trị rỗng (null).

IGNORE NULL: thuộc tính làm chỉ mục có thể nhận giá trị rỗng (null).

Ví dụ: Tạo chỉ mục (khóa) cho thuộc tính MaHV trên bảng Hocvien:

CREATE UNIQUE INDEX MaHocvien **ON** Hocvien(MaHV)

Xóa chỉ mục (Câu lệnh DROP INDEX)

Cú pháp:

DROP INDEX <tên chỉ mục> **ON** <bảng>

Ví dụ: Xóa chỉ mục (khóa) MaHocvien trên bảng Hocvien:

DROP INDEX MaHocvien **ON** Hocvien