

# Giới thiệu về ngôn ngữ SQL (phần 1)

Posts and Telecommunications Institute of Technology-PTIT



# Lịch sử của SQL

- SQL là viết tắt của từ tiếng Anh: Structural Query Language. SQL là một ngôn ngữ tuân thủ chuẩn cho việc tạo ra và truy vấn các CSDL quan hệ.
- SQL được chấp nhận bởi Viện tiêu chuẩn quốc gia Hoa Kỳ (ANSI) và tổ chức tiêu chuẩn quốc tế (ISO) cũng như thoả mãn các tiêu chuẩn xử lý thông tin của liên bang (FIPS).
- Giữa những năm 1974 và 1979, các nhân viên làm việc trong phòng nghiên cứu thí nghiệm của công ty IBM tại San Jose, bang California, Hoa Kỳ đã tiến hành phát triển hệ thống có tên là R, ngay sau khi bài báo truyền thống định nghĩa CSDL quan hệ được công bố. Mục tiêu của hệ thống R là chứng minh tính khả thi của việc cài đặt mô hình quan hệ trong một hệ quản trị CSDL. Họ sử dụng một ngôn ngữ có tên là SEQUEL (Structured English Query Language), là một ngôn ngữ nối tiếp của SQUARE (Specifying Queries as Relational Expressions). Cả hai đều được phát triển tại IBM, San Jose.
- Sau đó, SEQUEL được đổi tên thành SQL trong quá trình thực hiện dự án này.

# Lịch sử của SQL (cont.)

- Hệ thống R chưa từng được thương mại hóa nhưng nó trực tiếp dẫn đến sự phát triển của SQL/DS (bản SQL chạy trên hệ điều hành DOS năm 1981, và trên một phiên bản máy ảo VM năm 1982). Đây là hệ quản trị CSDL quan hệ được thương mại hóa đầu tiên của IBM.
- Tuy nhiên, IBM không phải là công ty đưa ra phiên bản cài đặt thương mại đầu tiên cho hệ quản trị CSDL quan hệ mà vinh dự đó thuộc về Oracle với phần mềm quan hệ năm 1979.
- Hiện nay, các hệ thống quản trị CSDL quan hệ đều dựa trên SQL.
- Mỗi nhà cung cấp dịch vụ đều có toàn bộ các đặc tính chuẩn của SQL, kèm theo đó là các tính năng phụ của riêng từng hãng. Các phần mở rộng trong SQL này dẫn đến hiện tượng trùng lặp khi ứng dụng SQL được cài đặt trên các hệ CSDL khác nhau. Và những phần mở rộng này đặc trưng riêng cho từng nhà cung cấp.

# Lịch sử của SQL (cont.)

- SQL-99 (hay còn gọi là SQL3) là phiên bản hiện thời của chuẩn ANSO dành cho SQL. Chuẩn này cũng được chấp thuận bởi ISO.
- Mặc dù có rất nhiều phiên bản của SQL, bản chất bên trong của nó mới là điều cần quan tâm. Dù với Oracle, Microsoft SQL Server, IBM's DB2, Microsoft Access, MySQL, hay bất kỳ hệ quản trị CSDL quan hệ tiên tiến nào khác, những thông tin trong bài này sẽ giúp bạn nhanh chóng tìm ra điểm chính của các hệ thống này.

# SQL

- SQL là một ngôn ngữ CSDL quan hệ đầy đủ. Nó bao gồm cả ngôn ngữ định nghĩa dữ liệu (DDL) và ngôn ngữ thao tác dữ liệu (DML).
- Cả hai ngôn ngữ dữ liệu của SQL đều được đề cập đến trong bài này.
- Nếu dùng Microsoft Access thì bạn không cần biết nhiều về DDL của SQL so với nếu bạn dùng Oracle 9i hay MySQL.

# Ký pháp cho câu lệnh SQL

Ký pháp	Mô tả
VIẾT HOA	từ khoá cần thiết cho câu lệnh SQL
<i>Viết nghiêng</i>	Một tham số do người dùng cung cấp- thường là cần thiết
{a   b   ... }	Một tham số bắt buộc, sử dụng một trong số danh sách lựa chọn
[...]	Một tham số tùy chọn - mọi thứ trong ngoặc vuông đều là tùy chọn
<i>tablename</i>	Tên của bảng
<i>column</i>	Tên của một thuộc tính trong bảng
<i>data type</i>	Một định nghĩa kiểu dữ liệu hợp lệ
<i>constraint</i>	Một định nghĩa ràng buộc hợp lệ
<i>condition</i>	Một biểu thức điều kiện hợp lệ - trả về giá trị đúng hoặc sai
<i>columnlist</i>	Một hoặc nhiều tên cột hoặc biểu thức được phân cách nhau bởi dấu phẩy
<i>tablelist</i>	Một hoặc nhiều tên bảng được phân cách nhau bởi dấu phẩy
<i>conditionlist</i>	Một hoặc nhiều biểu thức điều kiện được phân cách nhau bởi dấu phẩy
<i>expression</i>	Một giá trị đơn (ví dụ 76 or 'married') hoặc một công thức (ví dụ, price-10)

# Ngôn ngữ định nghĩa dữ liệu trong SQL

- Trước khi sử dụng một hệ CSDL quan hệ, bạn phải thực hiện 2 việc: (1) tạo một cấu trúc CSDL, (2) tạo các bảng lưu dữ liệu người sử dụng.
- Công việc thứ nhất liên quan đến việc tạo ra các tệp vật lý để lưu trữ dữ liệu. Hệ CSDL quan hệ tự động tạo ra các bảng định nghĩa dữ liệu và tạo ra hệ quản trị CSDL ngầm định (default database administrator - DBA).
  - Việc tạo ra các tệp vật lý đòi hỏi sự tương tác giữa hệ điều hành và hệ quản trị cơ sở dữ liệu. Vì vậy, tạo ra cấu trúc cơ sở dữ liệu là một đặc tính có sự khác nhau từ một hệ quản trị cơ sở dữ liệu này sang hệ khác.
- Với một ngoại lệ là có thể tạo ra cơ sở dữ liệu, hầu hết các nhà cung cấp hệ thống quản trị cơ sở dữ liệu sử dụng bản SQL có khác một chút với bản SQL chuẩn của ANSI. Mặc dù vậy nhưng cũng chỉ thỉnh thoảng bạn mới gặp những sự khác nhau nhỏ trong cú pháp của các câu lệnh SQL. Ví dụ, hầu hết để yêu cầu mọi câu lệnh SQL được kết thúc bởi dấu chấm phẩy (;) tuy nhiên một số bản cài đặt SQL không sử dụng dấu (;). Hầu hết những sự khác nhau chung về cú pháp sẽ được chỉ ra trong bài giảng này hoặc ít nhất cũng liệt kê những sự khác nhau do người viết nhận thức được.



# Tóm tắt các câu lệnh định nghĩa dữ liệu SQL

Câu lệnh hoặc lựa chọn	Mô tả
CREATE SCHEMA AUTHORIZATION	Tạo một lược đồ CSDL
CREATE TABLE	Tạo một bảng mới trong CSDL người dùng
NOT NULL	Ràng buộc đảm bảo một cột sẽ không có giá trị rỗng
UNIQUE	Ràng buộc đảm bảo một cột sẽ không có giá trị trùng lặp
PRIMARY KEY	Định nghĩa một khóa chính cho một bảng
FOREIGN KEY	Định nghĩa một khóa ngoại cho một bảng
DEFAULT	Định nghĩa một giá trị mặc định cho một cột (khi không nhập giá trị mới nào vào)
CHECK	Ràng buộc dùng để kiểm tra tính đúng đắn của dữ liệu trong cột
CREATE INDEX	Tạo một chỉ mục cho một bảng
CREATE VIEW	Tạo một tập con động cho hàng/cột từ một hoặc nhiều bảng
ALTER TABLE	Thay đổi định nghĩa của một bảng: thêm/xóa/cập nhật các thuộc tính hoặc ràng buộc
DROP TABLE	Xóa vĩnh viễn một bảng (cùng dữ liệu của nó) khỏi lược đồ CSDL
DROP INDEX	Xóa vĩnh viễn một chỉ mục
DROP VIEW	Xóa vĩnh viễn một khung nhìn

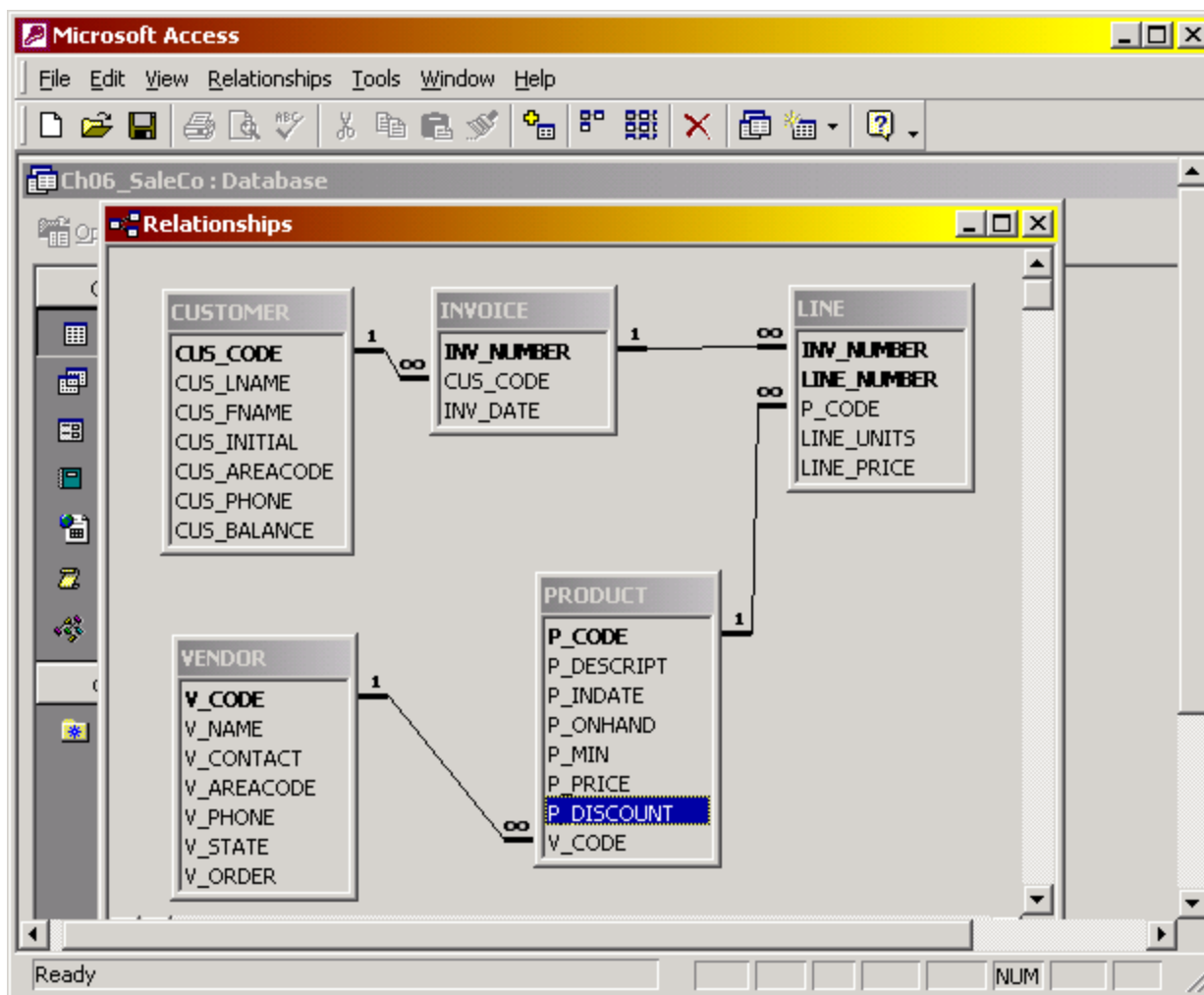


# Các câu lệnh của DDL trong SQL

- Chúng ta sẽ dùng cơ sở dữ liệu dưới đây để mô tả cho cách sử dụng của các lệnh DDL của SQL. Cơ sở dữ liệu này liên quan một chút tới cơ sở dữ liệu về nhà cung cấp - linh kiện - công việc - vận chuyển. Các quy định về nghiệp vụ của hệ thống này như sau:
  1. Một khách hàng có thể có nhiều yêu cầu trả tiền. mỗi bản yêu cầu trả tiền (invoice) chỉ được tạo ra bởi một khách hàng.
  2. Một invoice chứa một hoặc nhiều dòng. mỗi dòng của invoice liên quan tới một invoice.
  3. Mỗi dòng invoice là cho một mặt hàng. Một mặt hàng có thể tìm thấy ở nhiều dòng khác nhau. Một nhà cung cấp có thể cung cấp nhiều mặt hàng. Một vài nhà cung cấp có thể không cung cấp mặt hàng nào cả,
  4. Nếu một mặt hàng được cung cấp bởi một nhà cung cấp, thì mặt hàng đó chỉ được cung cấp bởi duy nhất nhà cung cấp đó.
  5. một số mặt hàng không được cung cấp bởi nhà cung cấp nào cả mà chúng được công ty tự sản xuất (in-house) hoặc được cung cấp qua những cách khác.



# Ví dụ về CSDL



# Tạo các cấu trúc bảng bằng SQL

- Câu lệnh CREATE TABLE có cú pháp như sau:

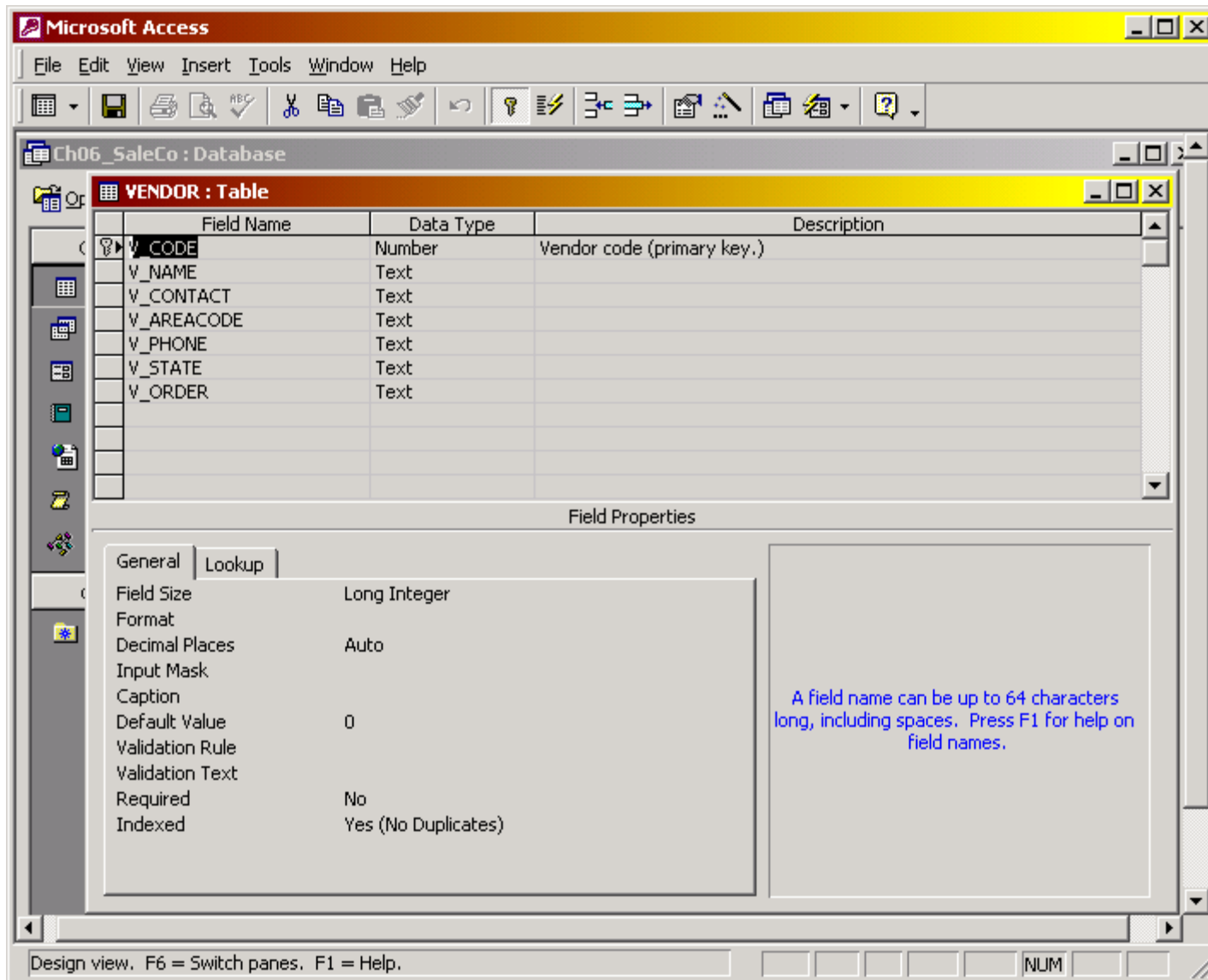
```
CREATE TABLE tablename (  
    column1    data type    [constraint] [,  
    column2    data type    [constraint] ] [,  
    PRIMARY KEY (column1 [,column2] ) ] [,  
    FOREIGN KEY (column1 [,column2] ) REFERENCES tablename ] [,  
    CONSTRAINT constraint ] ) ;
```

# Ví dụ – Tạo bảng

- Ví dụ tạo một bảng VENDOR của cơ sở dữ liệu ví dụ được mô tả ở trên.

```
CREATE TABLE VENDOR (  
    V_CODE          INTEGER          NOT NULL          UNIQUE,  
    V_NAME          VARCHAR(35)      NOT NULL,  
    V_CONTACT       VARCHAR(15)      NOT NULL,  
    V_AREACODE      CHAR(3)          NOT NULL,  
    V_PHONE         CHAR(8)          NOT NULL,  
    V_STATE         CHAR(2)          NOT NULL,  
    V_ORDER         CHAR(1)          NOT NULL,  
    PRIMARY KEY ( V_CODE));
```

# Bảng VENDOR trong Access

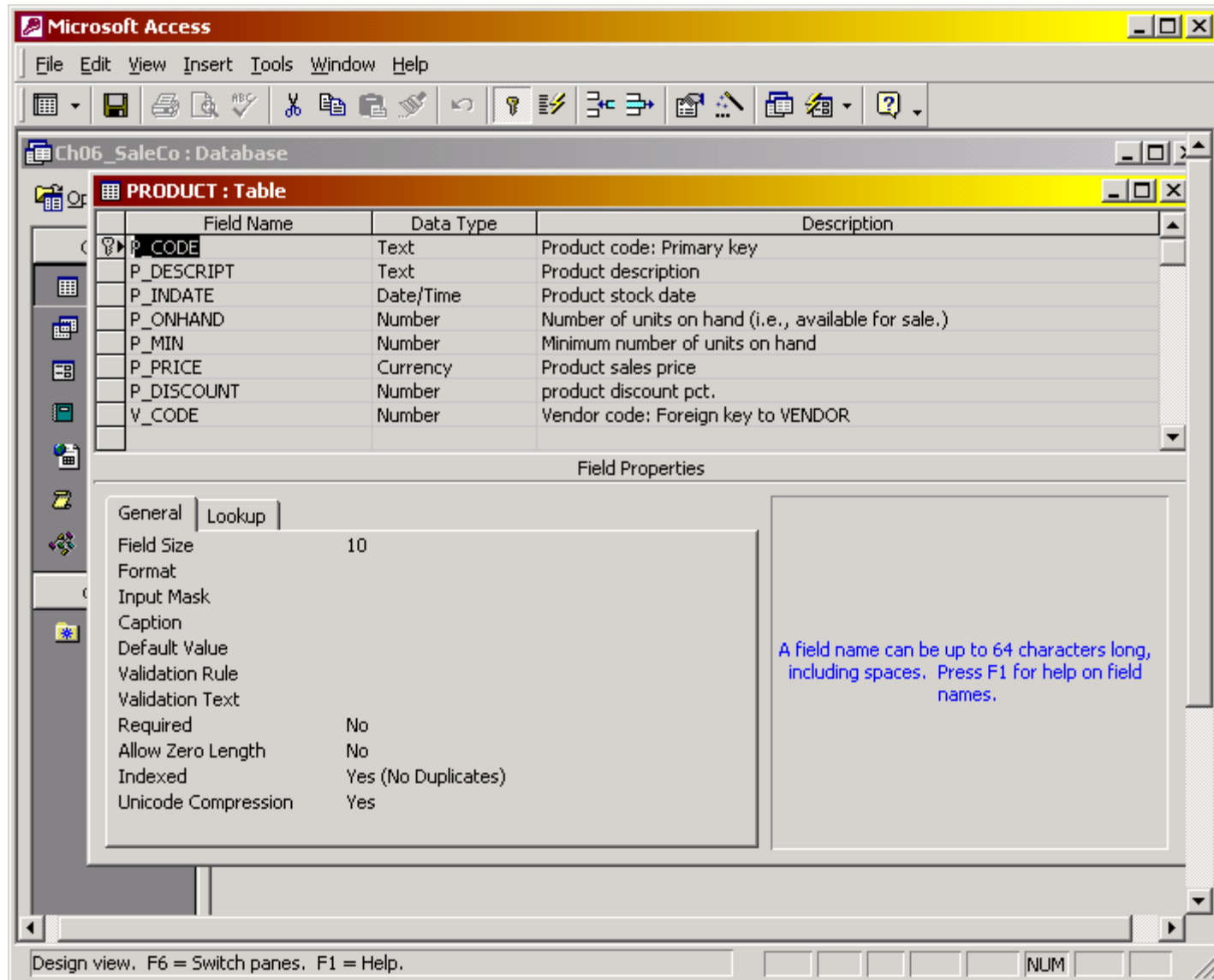


# Ví dụ – Tạo bảng

- Sau đó ta sẽ tạo tiếp đến bảng PRODUCT như sau:

```
CREATE TABLE PRODUCT (  
    P_CODE          VARCHAR(10)      NOT NULL          UNIQUE,  
    P_DESCRIPT      VARCHAR(35)      NOT NULL,  
    P_INDATE        DATE              NOT NULL,  
    P_ONHAND        SMALLINT          NOT NULL,  
    P_MIN           SMALLINT          NOT NULL,  
    P_PRICE         NUMBER(8,2)       NOT NULL,  
    P_DISCOUNT     NUMBER(4,2)       NOT NULL,  
    V_CODE          INTEGER,  
    PRIMARY KEY ( P_CODE),  
    FOREIGN KEY (V_CODE) REFERENCES VENDOR ON UPDATE CASCADE);
```

# Bảng PRODUCT trong Access



# Ví dụ – Tạo bảng


- Tạo ra bảng CUSTOMER như sau:

```
CREATE TABLE CUSTOMER (  
  CUS_CODE          NUMBER          PRIMARY KEY,  
  CUS_LNAME         VARCHAR(15)     NOT NULL,  
  CUS_FNAME         VARCHAR(15)     NOT NULL,  
  CUS_INITIAL       CHAR(1),  
  CUS_AREACODE      CHAR(3)         DEFAULT '615' NOT NULL  
                                     CHECK (CUS_AREACODE IN ('615', '713', '931')),  
  CUS_PHONE         CHAR(8)         NOT NULL,  
  CUS_BALANCE       NUMBER(9,2)     DEFAULT 0.00,  
  CONSTRAINT CUS_UI1 UNIQUE (CUS_LNAME, CUS_FNAME));
```


Ràng buộc  
cột



Ràng buộc  
bảng

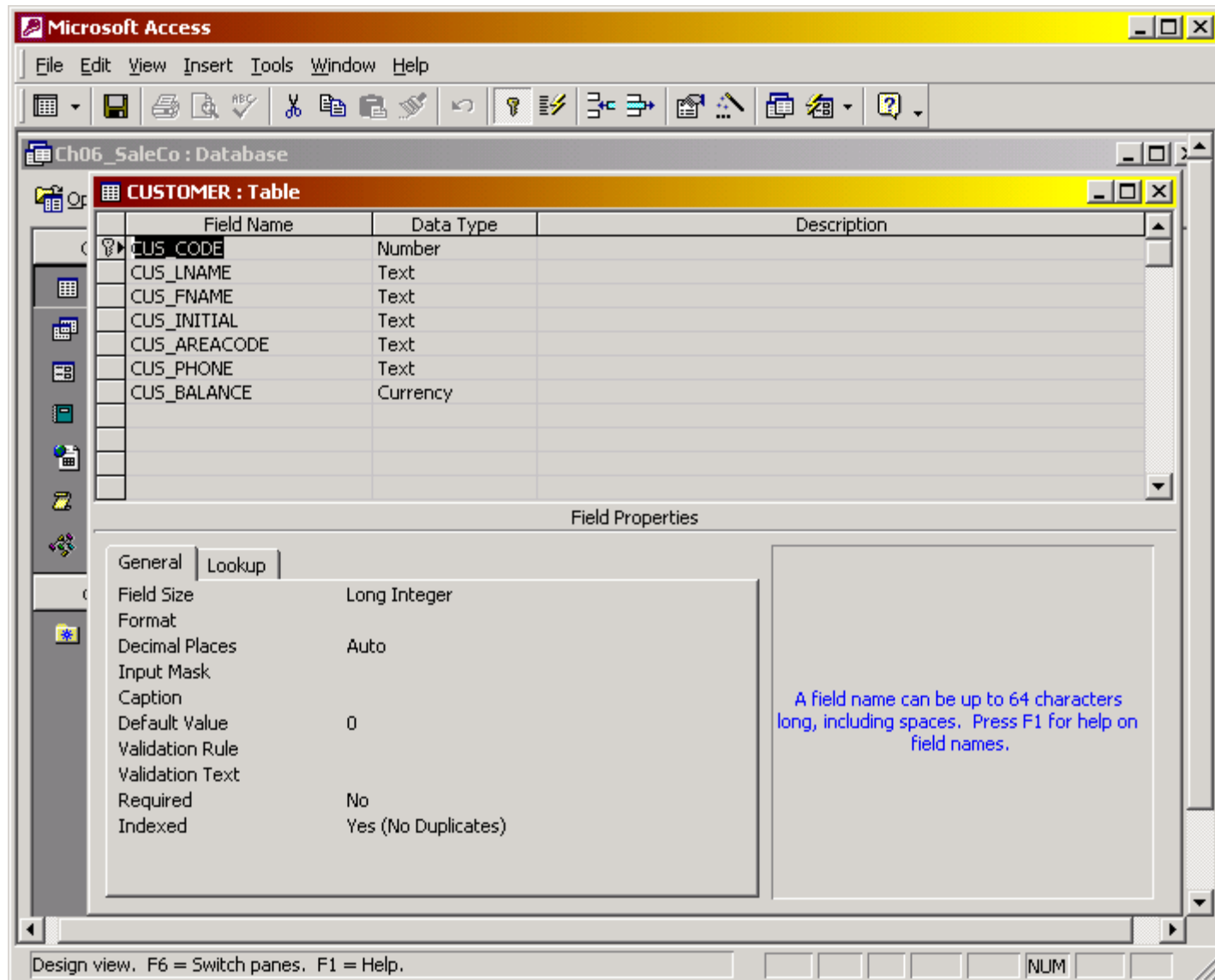


Tạo một ràng buộc chỉ mục duy nhất là CUS\_UI1 trên họ và tên của khách hàng.





# Bảng CUSTOMER trong Access



# Ví dụ – Tạo bảng

- Tạo bảng INVOICE như sau:

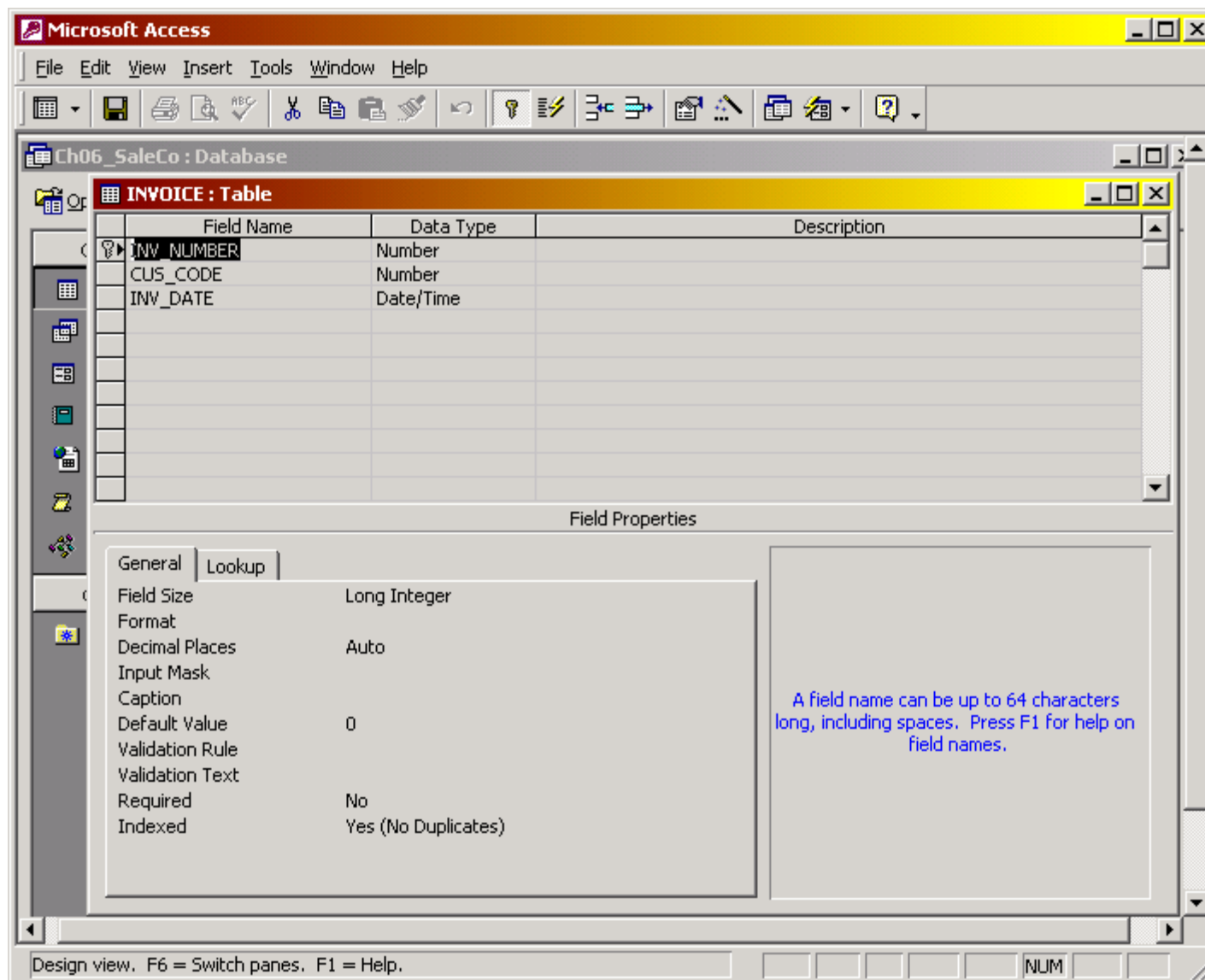
```
CREATE TABLE INVOICE (  
  INV_NUMBER      NUMBER          PRIMARY KEY,  
  CUS_CODE        NUMBER          NOT NULL, REFERENCES CUSTOMER(CUS_CODE)  
  INV_DATE        DATE            DEFAULT SYSDATE NOT NULL,  
  CONSTRAINT INV_CK1 CHECK (INV_DATE > TO_DATE('01-JAN-2002', 'DD-MON-YYYY')));
```

Một cách khác để định nghĩa khóa ngoại

Hàm đặc biệt để trả về ngày hiện tại

ràng buộc CHECK được sử dụng để kiểm tra tính hợp lệ của ngày invoice có lớn hơn 1/1/2002 không. Hàm TO\_DATE cần hai tham số bao gồm ngày cụ thể và định dạng ngày được sử dụng.

# Bảng INVOICE trong Access



# Ví dụ – Tạo bảng

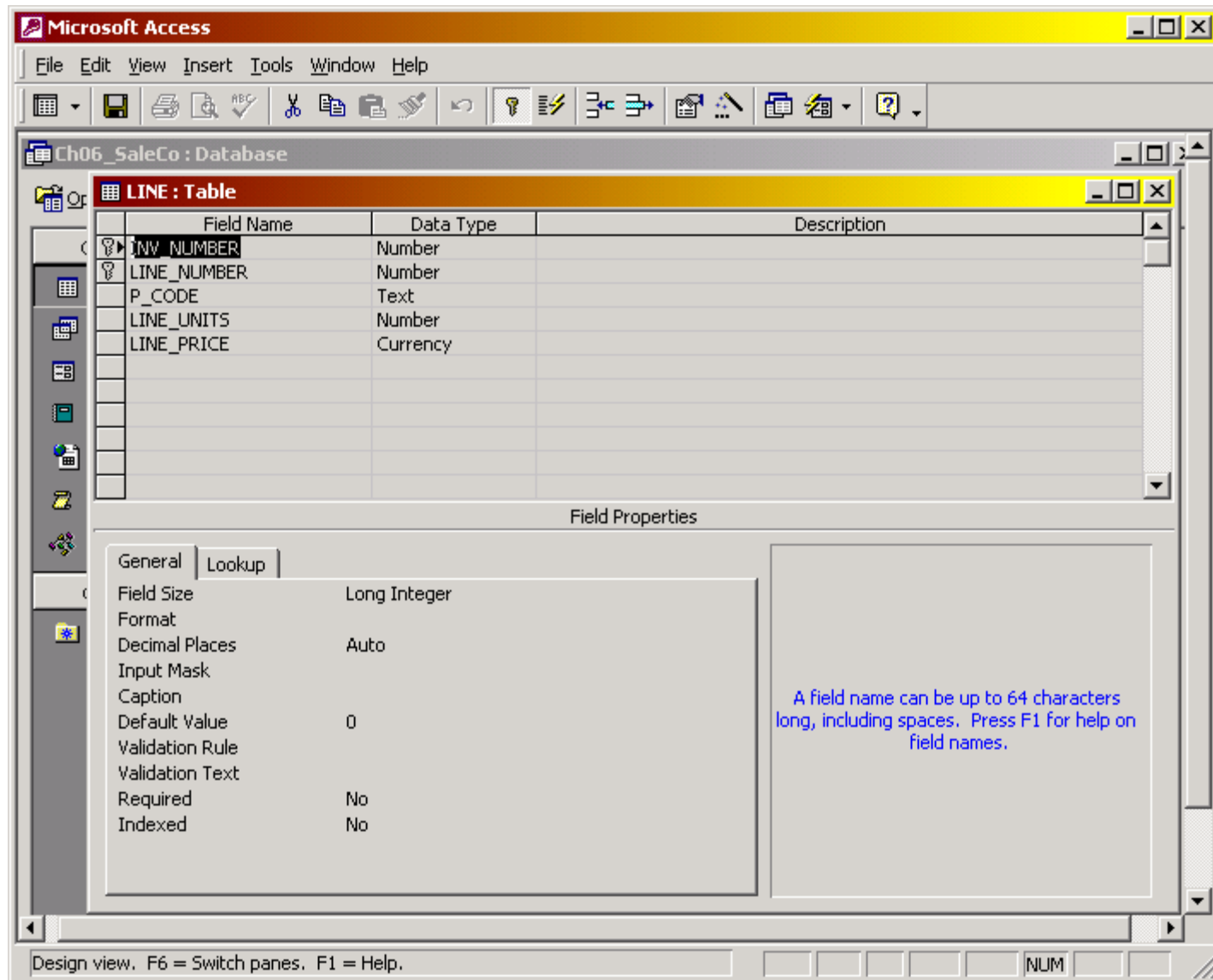
- Cuối cùng, tạo bảng LINE như sau:

```
CREATE TABLE LINE (  
    INV_NUMBER      NUMBER          NOT NULL,  
    LINE_NUMBER     NUMBER(2,0)     NOT NULL,  
    P_CODE           VARCHAR(10)     NOT NULL,  
    LINE_UNITS       NUMBER(9,2)     DEFAULT 0.00 NOT NULL,  
    LINE_PRICE       NUMBER(9,2)     DEFAULT 0.00 NOT NULL,  
    PRIMARY KEY (INV_NUMBER, LINE_NUMBER),  
    FOREIGN KEY (INV_NUMBER) REFERENCES INVOICE ON DELETE CASCADE  
    FOREIGN KEY (P_CODE) REFERENCES PRODUCT(P_CODE),  
    CONSTRAINT LINE_UI1 UNIQUE(INV_NUMBER, P_CODE));
```

việc sử dụng ON DELETE CASCADE được khuyến cáo nên dùng cho các thực thể yếu để đảm bảo rằng việc xoá một dòng trong thực thể chính sẽ gây ra việc xoá tự động các dòng tương ứng trong thực thể yếu phụ thuộc vào thực thể chính đó

ràng buộc trên toàn bảng để ngăn chặn việc có hai dòng trong một invoice giống nhau

# Bảng LINE trong Access



# Một số lưu ý trong việc tạo bảng

- Đối với cơ sở dữ liệu ví dụ trên, bảng PRODUCT chứa một khoá ngoại tham chiếu tới bảng VENDOR. Vì vậy, bảng VENDOR phải được tạo trước. Nói chung, các bảng nằm bên phía lực lượng 1 của một quan hệ 1-nhiều phải được tạo trước khi bảng bên phía lực lượng nhiều có thể được tạo ra.
- Với hệ thống Oracle9i nếu bạn sử dụng cách định nghĩa khoá chính bằng từ khoá PRIMARY KEY bạn không cần đưa yêu cầu NOT NULL và UNIQUE vào câu lệnh tạo bảng nữa. Thực tế, bạn sẽ nhận được một thông báo lỗi nếu bạn làm như vậy.
- Ràng buộc ON UPDATE CASCADE là một phần của chuẩn ANSI nhưng nhiều hệ quản trị cơ sở dữ liệu không hỗ trợ nps. Oracle là một trong số những hệ thống quản trị không hỗ trợ tính năng này.
- Nếu khoá chính là một khoá ghép, tất cả các thuộc tính của khoá được chứa trong một dấu ngoặc đơn và được phân tách nhau bởi dấu phẩy. Ví dụ, bảng LINE có khoá chính được định nghĩa như sau:

PRIMARY KEY (inv\_number, line\_number).



# Một số lưu ý trong việc tạo bảng (cont.)

- Hỗ trợ ràng buộc tham chiếu rất đa dạng, thay đổi từ hệ quản trị này sang hệ quản trị khác.
  - MS Access, SQL Server và Oracle hỗ trợ ON DELETE CASCADE.
  - MS Access, SQL Server hỗ trợ ON UPDATE CASCADE.
  - Oracle không hỗ trợ ON UPDATE CASCADE.
  - Oracle hỗ trợ SET NULL.
  - MS Access, SQL Server không hỗ trợ SET NULL.
- MS Access không hỗ trợ ON DELETE CASCADE hoặc ON UPDATE CASCADE tại mức câu lệnh SQL tuy nhiên nó lại hỗ trợ thông qua giao diện cửa sổ quan hệ.

# Các kiểu dữ liệu

1. INT or INTEGER.
2. REAL or FLOAT.
3. CHAR( $n$ ) = chuỗi ký tự có độ dài cố định.
4. VARCHAR( $n$ ) = chuỗi ký tự có độ dài thay đổi, và có tối đa  $n$  ký tự.
5. NUMERIC( $precision, decimal$ ) = kiểu số với độ dài  $precision$  số, và độ chính xác “ $decimal$ ” đơn vị sau dấu phẩy.  
NUMERIC(10,2) có thể chứa số lớn đến  $\pm 99,999,999.99$
6. DATE = ngày tháng. SQL có định dạng 'yyyy-mm-dd'
7. TIME = thời gian. SQL có định dạng 'hh:mm:ss[.ss...]'.
8. DATETIME or TIMESTAMP. SQL có định dạng TIMESTAMP 'yyyy-mm-dd hh:mm:ss[.ss...]'.



# Khai báo khóa

- Có thể dùng PRIMARY KEY hoặc UNIQUE
- SQL chỉ cho phép đánh chỉ số (index) bằng PRIMARY KEY.
- SQL không cho phép thuộc tính PRIMARY KEY chứa giá trị rỗng. Tuy nhiên, nó lại cho phép các thuộc tính UNIQUE mang giá trị rỗng (có thể có nhiều hơn một bản ghi mang giá trị rỗng, tuy nhiên các giá trị khác rỗng của các bản ghi khác nhau phải khác nhau).

VD:

```
CREATE TABLE Bán (  
    quán CHAR(20),  
    bia VARCHAR(20),  
    giá REAL,  
    PRIMARY KEY(quán, bia) );
```

```
CREATE TABLE Bán (  
    quán CHAR(20),  
    bia VARCHAR(20),  
    giá REAL,  
    UNIQUE(quán, bia) );
```



# Lưu ý khi khai báo khóa bằng UNIQUE

- Các câu lệnh sau là khác nhau.

```
CREATE TABLE Bán (  
    quán CHAR(20) UNIQUE,  
    bia VARCHAR(20) UNIQUE,  
    giá REAL,  
);
```

```
CREATE TABLE Bán (  
    quán CHAR(20),  
    bia VARCHAR(20),  
    giá REAL,  
    UNIQUE(quán, bia) );
```

# Khóa ngoại

```
CREATE TABLE Bia (  
    tên          CHAR(20) PRIMARY KEY,  
    Nhà_SX      CHAR(20)  
);
```

```
CREATE TABLE Bán (  
    quán CHAR(20),  
    bia CHAR(20) REFERENCES Bia(tên),  
    giá REAL  
);
```

Hoặc:

```
CREATE TABLE Bán (  
    quán CHAR(20),  
    bia CHAR(20),  
    giá REAL,  
    FOREIGN KEY bia REFERENCES Bia(tên)  
);
```

# Thiết lập các quy tắc

- **Thêm** ON [DELETE, UPDATE] [CASCADE, SET NULL] trong quá trình khai báo khóa ngoại.

Ví dụ:

```
CREATE TABLE Bán (  
    quán CHAR(20),  
    bia CHAR(20),  
    giá REAL,  
    FOREIGN KEY bia REFERENCES Bia(tên)  
        ON DELETE SET NULL  
        ON UPDATE CASCADE  
);
```

# Kiểm tra trên từng thuộc tính

- CHECK (*condition*) : kiểm tra giá trị của từng bản ghi tại thuộc tính nào đó theo điều kiện có trong *condition*.
- Các điều kiện trong *condition* chỉ được kiểm tra khi giá trị của các thuộc tính liên quan đến nó bị thay đổi (chèn, cập nhật).

Ví dụ:

```
CREATE TABLE Bán (  
    quán CHAR(20),  
    bia CHAR(20) CHECK(  
        bia IN (SELECT tên  
                FROM Bia) ),  
    giá REAL CHECK(  
        giá <= $5.00 )  
);
```

# Kiểm tra trên từng bản ghi

- Được khai báo riêng sau khi đã khai báo các thuộc tính,
- Điều kiện *condition* có thể liên quan đến bất kỳ thuộc tính nào trong bảng
- Kiểm tra trong quá trình thêm, sửa bản ghi.

Ví dụ:

```
CREATE TABLE Bán (  
    quán CHAR(20),  
    bia CHAR(20),  
    giá REAL,  
    CHECK (quán = 'Hải Xồm' OR giá <= $5.00)  
);
```

Chỉ quán Hải Xồm được phép bán bia đắt hơn \$5.00



# Một số đặc tính khác của thuộc tính

1. NOT NULL = mọi bản ghi phải có một giá trị nào đó tại thuộc tính này.
2. DEFAULT *value* = đặt mặc định một giá trị cho thuộc tính này trong trường hợp không nhập giá trị cho nó.

VD:

```
CREATE TABLE Khách_hàng (  
    tên CHAR(30) PRIMARY KEY,  
    địa_chỉ CHAR(50) DEFAULT '123 Sesame St',  
    Đ_thoại CHAR(16)  
);
```

- Khóa chính được mặc định là NOT NULL.



# Chèn giá trị mặc định

1. DEFAULT DATE/TIME/TIMESTAMP.
2. Tạo chuỗi số tự động bằng SEQUENCE

VD:

```
CREATE SEQUENCE Ma_KH;  
CREATE TABLE Khách_hang (  
    maKH INTEGER  
        DEFAULT nextval('Ma_KH'),  
    tên VARCHAR(30)  
);
```



# Các câu lệnh DDL nâng cao trong SQL

- Các câu lệnh SQL thay đổi cấu trúc của bảng bằng cách thay đổi đặc tính hoặc thêm các thuộc tính.
- Các câu lệnh liên quan đến việc thêm dữ liệu và các cột dữ liệu đã thay đổi cũng được đề cập.
- Cách nhân đôi hoặc xóa cả bảng hoặc từng phần của bảng.

# Câu lệnh ALTER TABLE

- Việc thay đổi cấu trúc của bảng được thực hiện bằng lệnh ALTER TABLE, kèm theo một từ khóa chỉ rõ việc thay đổi là gì.
- Có 3 từ khóa cơ bản được sử dụng: ADD, MODIFY, và DROP.
  - **ADD** cho phép thêm cột vào bảng.
  - **MODIFY** cho phép thay đổi đặc tính của bảng.
  - **DROP** cho phép xóa cột của bảng. Hầu hết các hệ CSDL quan hệ không cho phép xóa cột trừ khi không có giá trị nào trong cột muốn xóa vì nó liên quan mật thiết với các bảng dữ liệu khác.

# Câu lệnh ALTER TABLE (cont.)

- Cú pháp cơ bản của lệnh ALTER TABLE là:

```
ALTER TABLE tablename  
{ADD | MODIFY} ( columnname datatype  
[ {ADD | MODIFY} columnname datatype ] );
```

- Lệnh ALTER TABLE cũng được dùng để thêm ràng buộc cho bảng. Trường hợp này có cú pháp như sau:

```
ALTER TABLE tablename  
ADD constraint [ ADD constraint];
```

# Câu lệnh ALTER TABLE (cont.)

- Câu lệnh ALTER TABLE còn được dùng để xóa cột hoặc ràng buộc của bảng. Cú pháp cơ bản trong trường hợp này là:

```
ALTER TABLE tablename  
DROP { PRIMARY KEY |  
      COLUMN columnname |  
      CONSTRAINT constraintname } ;
```

- Lưu ý rằng khi xóa ràng buộc của bảng ta cần xác định tên của ràng buộc đó. Đây là lý do tại sao cần phải đặt tên cho các ràng buộc trong các lệnh CREATE TABLE hoặc ALTER TABLE.

# Thay đổi kiểu dữ liệu cho cột

- Câu lệnh ALTER TABLE cũng được dùng để đổi kiểu dữ liệu cho cột.
- Ví dụ, nếu ta cần phải đổi kiểu dữ liệu của V\_CODE trong bảng PRODUCT từ kiểu integer thành character. Câu lệnh SQL sẽ như sau:

```
ALTER TABLE PRODUCT  
MODIFY (V_CODE CHAR(5));
```

- Hầu hết các hệ CSDL quan hệ không cho phép thay đổi kiểu dữ liệu của các thuộc tính trừ khi thuộc tính đó là rỗng. Ví dụ, nếu ta chạy câu lệnh SQL nói trên trong CSDL mà ta đang dùng, dòng báo lỗi sẽ hiện ra bởi vì cột V\_CODE đang chứa dữ liệu. Lý do của lỗi này đơn giản bởi vì thuộc tính V\_CODE trong bảng PRODUCT chỉ đến cột V\_CODE trong bảng VENDOR. Nếu hai cột dữ liệu không cùng kiểu sẽ dẫn đến xung đột giá trị tham vấn. Nếu cột V\_CODE trong bảng PRODUCT là rỗng và khóa ngoại không được xác định trong lúc tạo bảng PRODUCT thì câu lệnh SQL nói trên sẽ được thực hiện đúng.

# Thay đổi đặc tính dữ liệu của cột

- Nếu cột cần thay đổi đang chứa dữ liệu, ta có thể thực hiện bất kỳ thay đổi nào miễn là không làm thay đổi kiểu dữ liệu của nó.
- Ví dụ, nếu cần tăng độ rộng của cột P\_PRICE từ 8 lên 9 ký tự, ta cần phải viết câu lệnh như sau:

```
ALTER TABLE PRODUCT  
MODIFY (P_PRICE DECIMAL(9,2));
```

- Nhiều hệ CSDL quan hệ giới hạn các thay đổi có thể thực hiện lên cột. Ví dụ, Oracle cho phép mở rộng các cột nhưng lại không cho thu hẹp chúng lại.

# Thêm cột vào bảng

- Một bảng CSDL có thể được thay đổi bằng cách thêm hoặc xóa các cột.
- Ví dụ, nếu ta cần thêm cột P\_SALECODE vào bảng PRODUCT, cột này cho phép ta kiểm tra xem những mặt hàng nào đã được kiểm kê về thời gian cần được bày bán. Giả sử các giá trị trong cột P\_SALECODE nằm trong tập {1, 2, 3}, và không có phép tính trong đó thì ta gán kiểu character cho cột này.

```
ALTER TABLE PRODUCT
ADD (P_SALECODE CHAR(1));
```

# Thêm cột vào bảng (cont.)

- Khi thêm cột, không được cho cụm từ NOT NULL vào cột mới. Điều này dẫn đến báo lỗi vì khi ta thêm cột vào bảng, các hàng có sẵn trong bảng sẽ mặc định giá trị là rỗng cho cột đó. Bởi vậy, ta không thể gán cụm từ NOT NULL cho cột này.
- Ta có thể thêm cụm từ NOT NULL vào cấu trúc của bảng sau khi toàn bộ dữ liệu của cột mới đã được nhập vào, và cột đó không chứa giá trị rỗng nào.



# Xóa cột từ một bảng

- Đôi khi việc xóa đi một vài cột trong bảng là cần thiết.
- Giả sử ta cần xóa cột V\_ORDER trong bảng VENDOR. Câu lệnh SQL sau đây có thể được dùng:

```
ALTER TABLE VENDOR  
DROP COLUMN V_ORDER;
```

- Tương tự như phần trước, một vài hệ CSDL quan hệ có các giới hạn trong việc xóa cột khỏi bảng. Ví dụ, hầu hết các hệ CSDL quan hệ không cho phép xóa các cột có quan hệ với khóa ngoại, cũng như xóa các cột chứa khóa đó.

# Thêm các chỉ định khóa chính và khóa ngoại

- Mặc dù ta có thể tạo một bảng mới dựa trên một bảng có sẵn như trong ví dụ vừa rồi, quá trình thực hiện vẫn có thể gặp rắc rối. Về cơ bản, bảng PART được tạo ra mà không cần bảo đảm tính toàn vẹn của CSDL cũ. Cụ thể, ta không cần phải gán khóa chính cho bảng ở slide trước.
- Để xác định khóa chính cho bảng này, ta dùng lệnh ALTER như sau:

```
ALTER TABLE PRODUCT  
ADD PRIMARY KEY (P_CODE);
```

## Thêm các chỉ định khóa chính và khóa ngoại (cont.)

- Thực tế rằng luật toàn vẹn không được tự động thừa kế từ bảng cũ sang bảng mới, do vậy trong nhiều trường hợp chúng ta không có sự toàn vẹn về thực thể cũng như các mối liên kết.
- Ví dụ, ta có thể quên xác định khóa chính và khóa ngoại trong quá trình tạo bảng.
- Các quy tắc toàn vẹn có thể được tái thiết lập sử dụng câu lệnh ALTER như sau:

```
ALTER TABLE PRODUCT  
  ADD PRIMARY KEY(P_CODE)  
  ADD FOREIGN KEY(V_CODE) REFERENCES VENDOR;
```

# Xóa bảng khỏi CSDL

- Ta có thể xóa bảng PRODUCT bằng lệnh DROP như sau:

```
DROP TABLE PRODUCT
```

- Một bảng chỉ có thể được xóa khỏi CSDL nếu nó không tham gia vào bất kỳ một mối quan hệ nào. Nếu ta cố xóa một bảng mà vẫn tham gia vào quan hệ thì sẽ có bản tin báo lỗi hiện ra.