Bài thực hành số 2

Các kiểu dữ liệu. Tạo và sửa đổi cấu trúc bảng

❖ Nội dung chính:

- Các kiểu dữ liệu của MySQL
- Tạo các bảng dữ liệu
- Thay đổi cấu trúc bảng
- Xóa bảng

1. Các kiểu dữ liệu

MySQL hỗ trợ các bảng CSDL chứa các cột với các kiểu dữ liệu khác nhau. Các bảng dưới đây liệt kê các kiểu dữ liệu MySQL hỗ trợ.

Các kiểu dữ liệu số

Bảng sau mô tả một các kiểu dữ liệu số trong MySQL:

Kiểu	**Lưu trữ**
TINYINT	1 byte
SMALLINT	2 bytes
MEDIUMINT	3 bytes
INT/INTEGER	4 bytes
BIGINT	8 bytes

Lưu ý: *Kiểu BOOLEAN tương ứng với TINYINT(1)*

Kiểu dữ liệu	**Lưu trữ**
FLOAT	4 bytes
DOUBLE	8 bytes
DECIMAL	Phụ thuộc vào khi định nghĩa cột

Các kiểu dữ liệu xâu

Trong MySQL, xâu có thể lưu mọi thứ từ dữ liệu văn bản tới dữ liệu nhị phân như ảnh, file. Xâu có thể được so sánh và tìm kiếm dựa trên mẫu sử dụng mệnh đề LIKE hoặc biểu thức chính quy. Bảng phía dưới là các kiểu dữ liệu xâu trong MySQL:

Kiểu dữ liệu xâu	**Mô tả**

CHAR	Một chuỗi ký tự có độ dài cố định
VARCHAR	Một chuỗi ký tự có độ dài có thể thay đổi
BINARY	Một chuỗi nhị phân độ dài có định
VARBINARY	Một chuỗi nhị phân độ dài có thể thay đổi
TINYBLOB	Một đối tượng nhị phân rất nhỏ
BLOB	Một đối tượng nhị phân nhỏ
MEDIUMBLOB	Một đối tượng nhị phân cỡ trung bình
LONGBLOB	Một đối tượng nhị phân cỡ lớn
TINYTEXT	Mỗi chuỗi văn bản rất nhỏ
TEXT	Mỗi chuỗi văn bản nhỏ
MEDIUMTEXT	Mỗi chuỗi văn bản cỡ trung bình
LONGTEXT	Mỗi chuỗi văn bản rất dài

Các kiểu dữ liệu ngày và thời gian

MySQL cung cấp kiểu dữ liệu ngày, thời gian và tổ hợp ngày và thời gian. Ngoài ra MySQL cũng cung cấp kiểu dữ liệu timestamp để lưu thời gian thay đổi của bản ghi.

Các kiểu dữ liệu	**Mô tả**
DATE	Giá trị ngày trong định dạng 'YYYY-MM-DD'
TIME	Giá trị thời gian trong định dạng 'hh:mm:ss'
DATETIME	Giá trị ngày tháng và thời gian trong định dạng 'YYYY-MM-DD hh:mm:ss'

TIMESTAMP	Giá trị nhãn thời gian trong định dạng 'YYYY-MM-DD hh:mm:ss'

Cột có kiểuTIMESTAMP đóng vai trò đặt biệt do được tự động cập nhật giá trị thời gian thay đổi gần nhất khi bản ghi được thêm vào hoặc cập nhật.

2. Tạo bảng Cơ sở dữ liệu

Để tạo bảng, MySQL sử dụng câu lệnh **CREATE TABLE**. Câu lệnh có cấu trúc như sau:

```
CREATE TABLE [IF NOT EXISTS] table_name(

            <column name><type> [<default value>] [column
                                              constraints],

     ...

           <column name><type> [<default value>] [column
                                             constraints],

     <table

     constraint>,

     ...

     <table constraint>


) type=table_type
```

MySQL hỗ trợ tùy chọn IF NOT EXISTS để tránh lỗi tạo bảng đã tồn tại trong CSDL *table_name* là tên bảng muốn tạo.

Giá trị DEFAULT: MySQL cho phép gán giá trị ngầm định cho một cột. Nếu giá trị của cột đó không được xác định khi thêm dữ liệu vào bảng, giá trị cột sẽ được gán giá trị *value*. Giá trị ngầm định của một cột là NULL.

Table_type : xác định kiểu của bảng dữ liệu khi lưu trữ (chú ý thuộc tính này là đặc điểm riêng của MySQL). Nếu không xác định thì MySQL sẽ sử dụng kiểu bảng ngầm định. MySQL hỗ trợ các kiểu bảng lưu trữ khác nhau, cho phép tối ưu CSDL theo mục đích sử dụng. Một số kiểu bảng trong MySQL như MyISAM, InnoDB, BerkeleyDB (BDB), MERGE, HEAP…

MyISAM: Các bảng MyISAM làm việc rất nhanh, nhưng không hỗ trợ giao dịch. Thường được sử dụng trong các ứng dụng Web, là kiểu bảng ngầm định trong các phiên bản MySQL trước 5.5

InnoDB: Các bảng InnoDB hỗ trợ giao dịch an toàn, hỗ trợ khóa ngoài. InnoDB là kiểu lưu trữ ngầm định từ phiên bản MySQL 5.5.

Định nghĩa tập các cột: Các cột được liệt kê với các thuộc tính như kiểu dữ liệu, giá trị ngầm định nếu có, các ràng buộc trên cột.

Các ràng buộc trong SQL gồm có: **Primary Key, Foreign Key, Not Null, Unique, Check.** Nếu dữ liệu cập nhật vi phạm ràng buộc đã khai báo sẽ bị từ chối.

Các ràng buộc có thể được định nghĩa theo hai cách:
1) *Column constraint* (Ràng buộc cột): ràng buộc được áp dụng cho một cột cụ thể

2) *Table constraint*(Ràng buộc bảng): được khai báo tách rời, có thể áp dụng ràng buộc cho một hoặc nhiều cột.

PRIMARY KEY (ràng buộc khóa chính): Ràng buộc này định nghĩa một cột hoặc một tổ hợp các cột xác định duy nhất mỗi dòng trong bảng

NOT NULL: Ràng buộc này yêu cầu giá trị của cột không được phép là NULL

UNIQUE: ràng buộc yêu cầu các giá trị của cột là phân biệt. Chú ý với ràng buộc này giá trị của cột có thể là NULL nếu ràng buộc NOT NULL không được áp dụng trên cột.

CHECK:

Ràng buộc khóa chính khai báo theo kiểu ràng buộc mức cột

```
Column_name datatype [CONSTRAINT constraint_name]
PRIMARY
KEY
```

Ràng buộc khóa chính khai báo theo kiểu ràng buộc mức bảng

```
[CONSTRAINT constraint_name] PRIMARY KEY
(column_name1, column_name2,..)
```

Ví dụ: Tạo bảng *employees* với khóa chính xác định khi định nghĩa cột

```
 CREATE TABLE employees (

employeeNumber int(11) NOT NULL PRIMARY KEY ,

lastName varchar(50) NOT NULL,

firstName varchar(50) NOT NULL,

extension varchar(10) NOT NULL,

email varchar(100) NOT NULL,

officeCode varchar(10) NOT NULL,


              reportsTo int(11) default NULL,

jobTitle varchar(50) NOT NULL

            ) ENGINE=InnoDB DEFAULT
CHARSET=utf8;
```

Hoặc sử dụng cách như trên và đặt tên cho ràng buộc đó

```
 CREATE TABLE employees (

              employeeNumber int(11) NOT NULL CONSTRAINT
 emp_id_pk    PRIMARY KEY,

lastName varchar(50) NOT NULL,

firstName varchar(50) NOT NULL,

extension varchar(10) NOT NULL,

email varchar(100) NOT NULL,

officeCode varchar(10) NOT NULL,

reportsTo int(11) default NULL,

jobTitle varchar(50) NOT NULL,

PRIMARY KEY (employeeNumber)
```

```
            ) ENGINE=InnoDB DEFAULT
CHARSET=utf8;
```

Đặt tên ràng buộc

Khai báo `CONSTRAINT <name> <constraint>` dùng để đặt tên ràng buộc. Mục đích của việc đặt tên ràng buộc là khi cập nhật dữ liệu vi phạm ràng buộc, hệ quản trị CSDL thường bao gồm tên ràng buộc vào thông báo lỗi. Ngoài ra có thể sử dụng tên ràng buộc khi sửa đổi hóa xóa ràng buộc. Như ở ví dụ trên, ràng buộc khóa chính được đặt tên là `emp_id_pk`.

Ví dụ: Tạo bảng *employees* với khóa chính xác định theo kiểu *ràng buộc bảng* thay vì khai báo cùng với định nghĩa cột.

```
 CREATE TABLE employees (

employeeNumber int(11) NOT NULL,

lastName varchar(50) NOT NULL,

firstName varchar(50) NOT NULL,

extension varchar(10) NOT NULL,

email varchar(100) NOT NULL,

officeCode varchar(10) NOT NULL,

reportsTo int(11) default NULL,

jobTitle varchar(50) NOT NULL,

PRIMARY KEY (employeeNumber)

            ) ENGINE=InnoDB DEFAULT
CHARSET=utf8;
```

FOREIGN KEY (Ràng buộc khóa ngoài)

Từ khóa FOREIGN KEY được dùng để xác định khóa ngoài. Trong ví dụ dưới xác định cột *country_id* làm khóa ngoài, tham chiếu đến khóa chính của bảng *country*.

```
CREATE TABLE city (      city_id SMALLINT
UNSIGNED NOT NULL AUTO_INCREMENT,   city
VARCHAR(50) NOT NULL,    country_id SMALLINT
UNSIGNED NOT NULL,

     last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
     ON UPDATE CURRENT_TIMESTAMP,

     PRIMARY KEY(city_id),

     CONSTRAINT fk_city_country FOREIGN KEY (country_id)
     REFERENCES country (country_id) ON DELETE RESTRICT
     ON UPDATE CASCADE

)
```

Ý nghĩa của các tùy chọn đi kèm khi khai báo ràng buộc khóa ngoài:

- `ON DELETE RESTRICT`: có nghĩa không cho phép xóa dòng dữ liệu ở bảng được tham chiếu khi còn dữ liệu tham chiếu tới. Trong ví dụ trên không được phép xóa dòng dữ liệu của bảng *country* nếu tồn tại dòng dữ liệu từ bảng *city* tham chiếu tới.

- `ON UPDATE CASCADE`: có nghĩa khi cập nhật dữ liệu ở bảng được tham chiếu, dữ liệu bên bảng tham chiếu sẽ được tự động cập nhật. Trong ví dụ trên, khi thay đổi dữ liệu của cột country_id của bảng country thì cột country_id của bảng city sẽ được tự động cập nhật.

- Khi không sử dụng các tùy chọn này, ngầm định RESTRICT sẽ được sử dụng cho các sự kiện DELETE và UPDATE.

Sau khi đã tạo các bảng dữ liệu, có thể kiểm tra xem cấu trúc của các cột dữ liệu trong

Ví dụ: Hiển thị thông tin của bảng *employees*

```
DESCRIBE employees;
```

Kết quả trả về từ MySQL server

Field	Type	Null	Key	Default
employeeNumber	int(11)	NO	PRI	NULL
lastName	varchar(50)	NO		NULL
firstName	varchar(50)	NO		NULL
extension	varchar(10)	NO		NULL
email	varchar(100)	NO		NULL
officeCode	varchar(10)	NO		NULL
reportsTo	int(11)	YES		NULL
jobTitle	varchar(50)	NO		NULL

Bên cạnh lệnh DESCRIBE có thể sử dụng câu lệnh:

```
SHOW CREATE TABLE Table_Name
```

sẽ hiển thị về câu lệnh được sử dụng để tạo ra bảng dữ liệu.

3. Thay đổi cấu trúc bảng

Bên cạnh tạo bảng, để sửa đổi cấu trúc bảng đã tồn tại trong CSDL sử dụng câu lệnh ALTER TABLE. Câu lệnh có thể được dùng để:

- Thêm, xóa, sửa các cột của bảng
- Thêm và xóa các ràng buộc

Cú pháp của lệnh ALTER TABLE như sau:

```
 ALTER TABLE table_name tùy chọn[, tùy chọn...]

Các tùy chọn:

      ADD [COLUMN] <column_definition>

     MODIFY [COLUMN] <create_definition>

      DROP [COLUMN] <column_name>

      ADD <table_constraint>

      DROP <constraint_name>
```

Ví dụ: Thêm cột *salary* có kiểu INT, không vượt quá 10 chữ số, ràng buộc không được để trống vào bảng dữ liệu employees

```
ALTER TABLE employees ADD salary INT(10) NOT NULL
```

Field	Type	Null	Key	Default
employeeNumber	int(11)	NO	PRI	NULL
lastName	varchar(50)	NO		NULL
firstName	varchar(50)	NO		NULL
extension	varchar(10)	NO		NULL
email	varchar(100)	NO		NULL
officeCode	varchar(10)	NO		NULL
reportsTo	int(11)	YES		NULL
jobTitle	varchar(50)	NO		NULL
salary	int(10)	YES		NULL

Ví dụ: Sửa kiểu của cột salary thành kiểu decimal(15,2)

```
ALTER TABLE employees MODIFY salary decimal(15,2);
```

Field	Type	Null	Key	Default
employeeNumber	int(11)	NO	PRI	NULL
lastName	varchar(50)	NO		NULL
firstName	varchar(50)	NO		NULL
extension	varchar(10)	NO		NULL
email	varchar(100)	NO		NULL
officeCode	varchar(10)	NO		NULL
reportsTo	int(11)	YES		NULL
jobTitle	varchar(50)	NO		NULL
salary	decimal(15,2)	YES		NULL

Ví dụ: Xóa cột *officeCode* khỏi bảng *employees*

```
ALTER TABLE employees DROP officeCode
```

Field	Type	Null	Key	Default
employeeNumber	int(11)	NO	PRI	NULL
lastName	varchar(50)	NO		NULL
firstName	varchar(50)	NO		NULL
extension	varchar(10)	NO		NULL
email	varchar(100)	NO		NULL
reportsTo	int(11)	YES		NULL
jobTitle	varchar(50)	NO		NULL
salary	decimal(15...	YES		NULL

4. Xóa bảng

Để xóa bảng khỏi CSDL, sử dụng câu lệnh DROP TABLE:

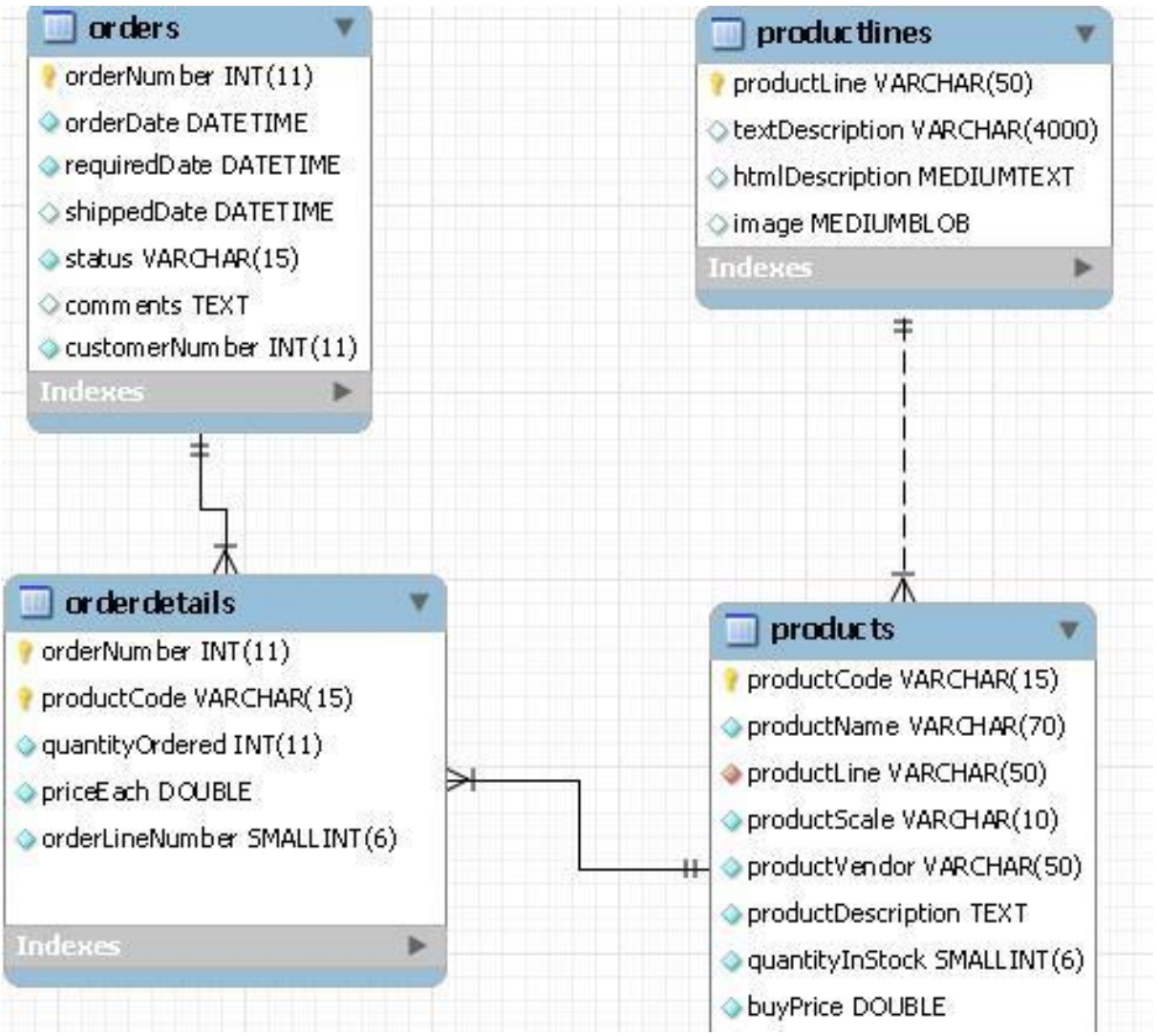
```
DROP TABLE [IF EXISTS] <table_name>
```

MySQL cho phép xóa nhiều bảng cùng lúc bằng cách liệt kê tên các bảng cách nhau bởi dấu phẩy. Tùy chọn IF EXISTS được sử dụng để tránh xóa bảng không tồn tại trong CSDL.

❖ **Bài tập thực hành**

1. Tạo CSDL my_classicmodels gồm 4 bảng: productlines, products, orders và orderdetails với các thuộc tính như trong hình vẽ phía dưới. Các khóa chính có kiểu INT sử dụng kiểu tự tăng AUTO_INCREMENT. *Gợi ý*: Khóa chính được tạo thành từ tổ hợp các cột cần khai báo theo ràng buộc mức bảng.

2. Sau khi đã tạo 4 bảng dữ liệu trên, thêm các ràng buộc khóa ngoài giữa các bảng như trong hình vẽ. Các ràng buộc khóa ngoài sử dụng thêm tùy chọn
   ON UPDATE
   CASCADE


orders
orderNumber INT(11)
orderDate DATETIME
requiredDate DATETIME
shippedDate DATETIME
status VARCHAR(15)
comments TEXT
customerNumber INT(11)
Indexes
productlines
productLine VARCHAR(50)
textDescription VARCHAR(4000)
htmlDescription MEDIUMTEXT
image MEDIUMBLOB
Indexes
orderdetails
orderNumber INT(11)
productCode VARCHAR(15)
quantityOrdered INT(11)
priceEach DOUBLE
orderLineNumber SMALLINT(6)
Indexes
products
productCode VARCHAR(15)
productName VARCHAR(70)
productLine VARCHAR(50)
productScale VARCHAR(10)
productVendor VARCHAR(50)
productDescription TEXT
quantityInStock SMALLINT(6)
buyPrice DOUBLE