TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

KHOA CÔNG NGHỆ THÔNG TIN

----------oOo----------

BÀI GIẢNG

THỰC HÀNH CƠ SỞ DỮ LIỆU

Giảng viên:

ThS. Vũ Bá Duy

ThS. Dư Phương Hạnh

ThS. Lê Hồng Hải

Hà Nội, Năm 2012

Lời nói đầu

Hiện nay có rất nhiều phần mềm Hệ quản trị cơ sở dữ liệu theo mô hình quan hệ (Relational DBMS) khác nhau, nhưng rất may mắn là các hệ quản trị cơ sở dữ liệu này sử dụng chung một ngôn ngữ được gọi là SQL (Structured Query Language- Ngôn ngữ truy vấn có cấu trúc). Các hệ quản trị cơ sở dữ liệu hiện nay đều cơ bản hỗ trợ chuẩn ANSI 2003 SQL.

Có thể nói ngôn ngữ SQL là một yếu tố đóng góp cho sự thành công của cơ sở dữ liệu quan hệ. Đây là ngôn ngữ mức cao nên người dùng chỉ cần viết lệnh thực hiện để đạt kết quả của truy vấn, phần tính toán và tối ưu hóa câu lệnh được hệ quản trị đảm nhận.

SQL bao gồm ba phần chính:

- Ngôn ngữ thao tác dữ liệu (Data manipulation language - DML): được sử dụng để lưu trữ, sửa đổi và truy xuất dữ liệu từ CSDL. Có những thành phần tiêu chuẩn dùng để thêm, cập nhật và xóa dữ liệu delete data.
- Ngôn ngữ định nghĩa dữ liệu (Data definition language - DDL): được sử dụng để định nghĩa cấu trúc của dữ liệu. Các câu lệnh này dùng để định nghĩa cấu trúc của cơ sở dữ liệu, bao gồm định nghĩa các hàng, các cột, các bảng dữ liệu, các chỉ số và một số thuộc tính khác liên quan đến cơ sở dữ liệu
- Ngôn ngữ điều khiển dữ liệu (Data control language - DCL): được sử dụng để quản lý truy cập tới dữ liệu của người dùng.

Nội dung các bài thực hành sẽ tập trung chủ yếu vào hai phần ngôn ngữ là DML và DDL và sử dụng DBMS mã nguồn mở MySQL server 5.5 làm công cụ thực hành. Nội dung trong các bài giảng chủ yếu là các thao tác, câu lệnh truy vấn, khai thác dữ liệu minh họa phần lý thuyết của môn học mà không nhằm tới việc sử dụng hay khai thác toàn bộ Hệ quản trị cơ sở dữ liệu MySQL.

Bài giảng "Thực hành cơ sở dữ liệu" gồm 10 bài thực hành; mỗi bài đều có 2 phần, phần thứ nhất: giới thiệu tóm tắt các khái niệm hoặc các câu lệnh cần thiết của bài giảng, phần thứ 2 là các bài tập thực hành sinh viên cần thực hiện dưới sự hướng dẫn trực tiếp của giáo viên hoặc tự thực hiện như các bài tập để củng cố nội dung của bài giảng.

Các yêu cầu trong suốt các bài thực hành được thao tác trên một Cơ sở dữ liệu mẫu. Các câu lệnh, ví dụ được thực hiện thống nhất trên MySQL 5.5.

Bài thực hành số 1

Cài đặt hệ quản trị CSDL và quản lý CSDL

❖ Nội dung chính

- Cài đặt MySQL server, thiết lập cổng làm việc, tạo tài khoản quản lý; kết nối với MySQL server.
- Cấu trúc thư mục của MySQL, ý nghĩa của từng thư mục.
- Làm quen với thao tác tạo cơ sở dữ liệu.

1. Cài đặt hệ quản trị CSDL MySQL Server

MySQL Server có thể chạy trên nhiều nền tảng khác nhau như Linux, Windows, Mac, FreeBSD, Unix. MySQL Server được cài đặt từ bản cài đặt hoặc được cài đặt bằng bản được biên dịch từ mã nguồn mở. MySQL Server có thể tải về từ địa chỉ http://dev.mysql.com/downloads/mysql/. Phần tiếp theo minh họa quá trình cài đặt trên hệ điều hành MS Windows.

Cài đặt trên hệ điều hành MS Windows

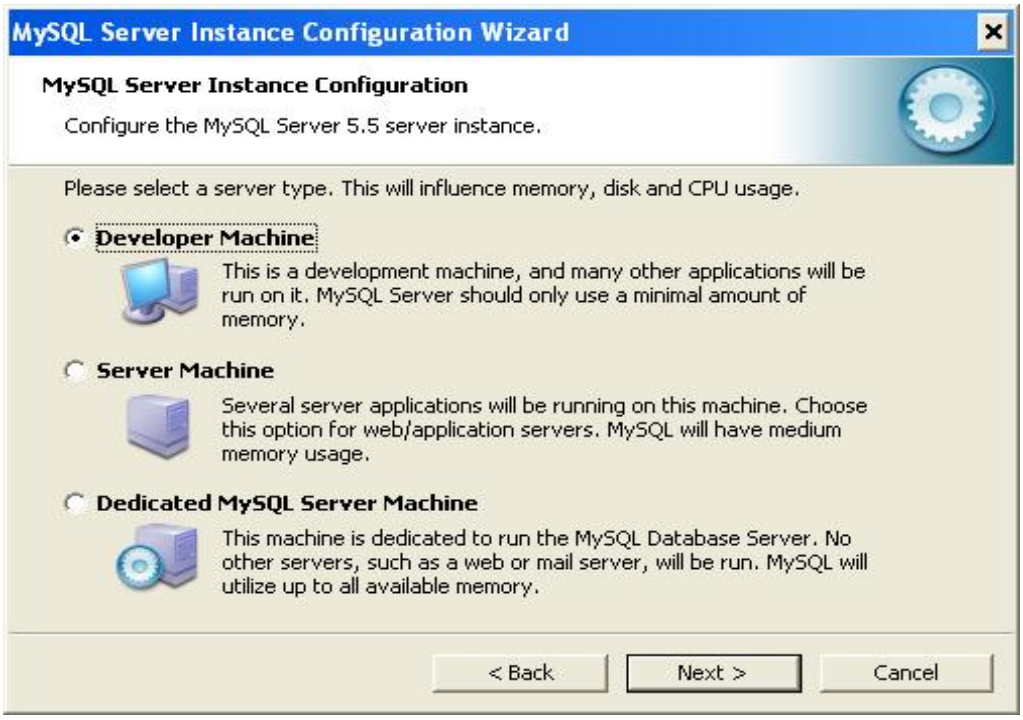
Sau khi thực hiện trình cài đặt trên Window, quá trình cài đặt MySQL Server bắt đầu qua các bước sau:

Bước 1: Lựa chọn kiểu server

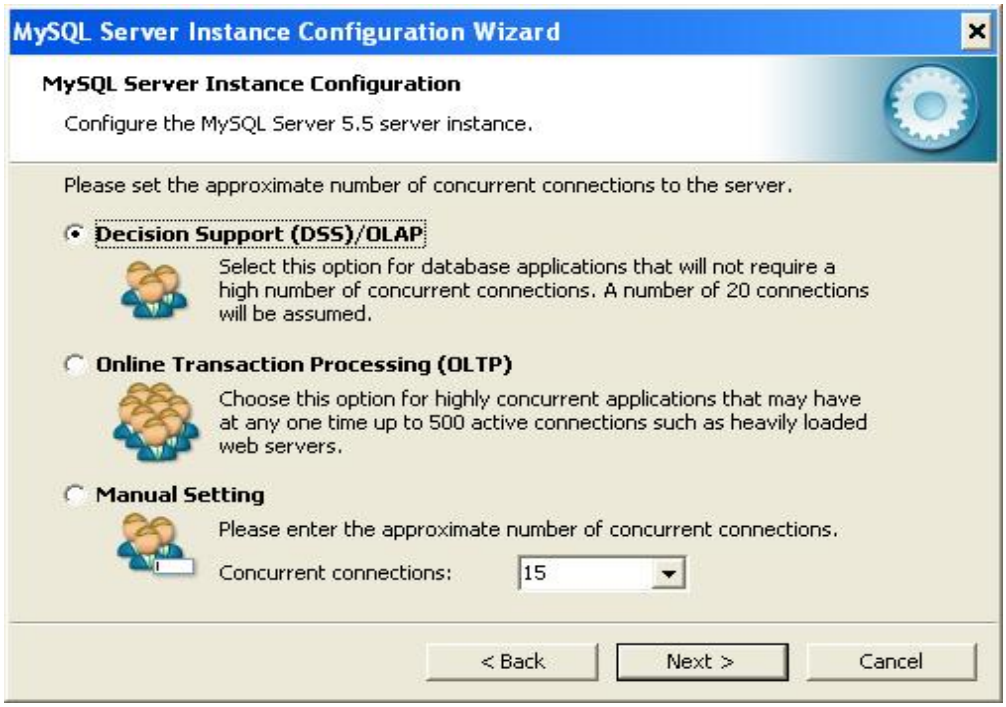
Chúng ta có thể lựa chọn 1 trong 3 kiểu server sau:

- Developer Machine: Lựa chọn này thích hợp khi cài đặt làm máy phát triển. Với cấu hình này, MySQL sẽ sử dụng số lượng bộ nhớ tối thiểu.
- Server Machine: Lựa chọn này thích hợp với máy tính chạy một số ứng dụng server như web/application server. MySQL sẽ sử dụng bộ nhớ trung bình trong cấu hình này.

- Dedicated MySQL Server Machine: Thích hợp cho máy tính chủ yếu làm server cơ sở dữ liệu (Database Server). Trong cấu hình này, MySQL sẽ sử dụng tối đa số lượng bộ nhớ của hệ thống.



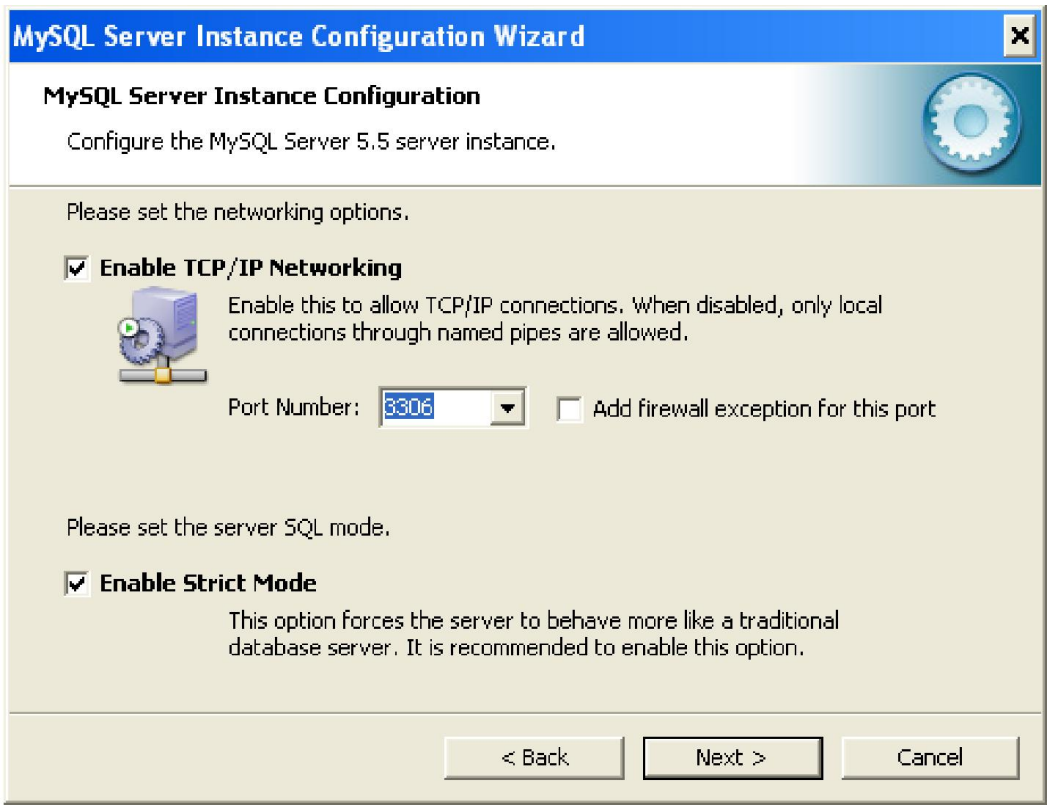
Bước 2: Cấu hình số lượng kết nối đồng thời



- Decision Support: thích hợp với ứng dụng không yêu cầu số lượng kết nối đồng thời cao
- OLTP: thích hợp với ứng dụng yêu cầu số lượng kết nối đồng thời cao, như webserver có tải lớn.
- Manual Setting: cho phép người sử dụng tự thiết lập số kết nối đồng thời.

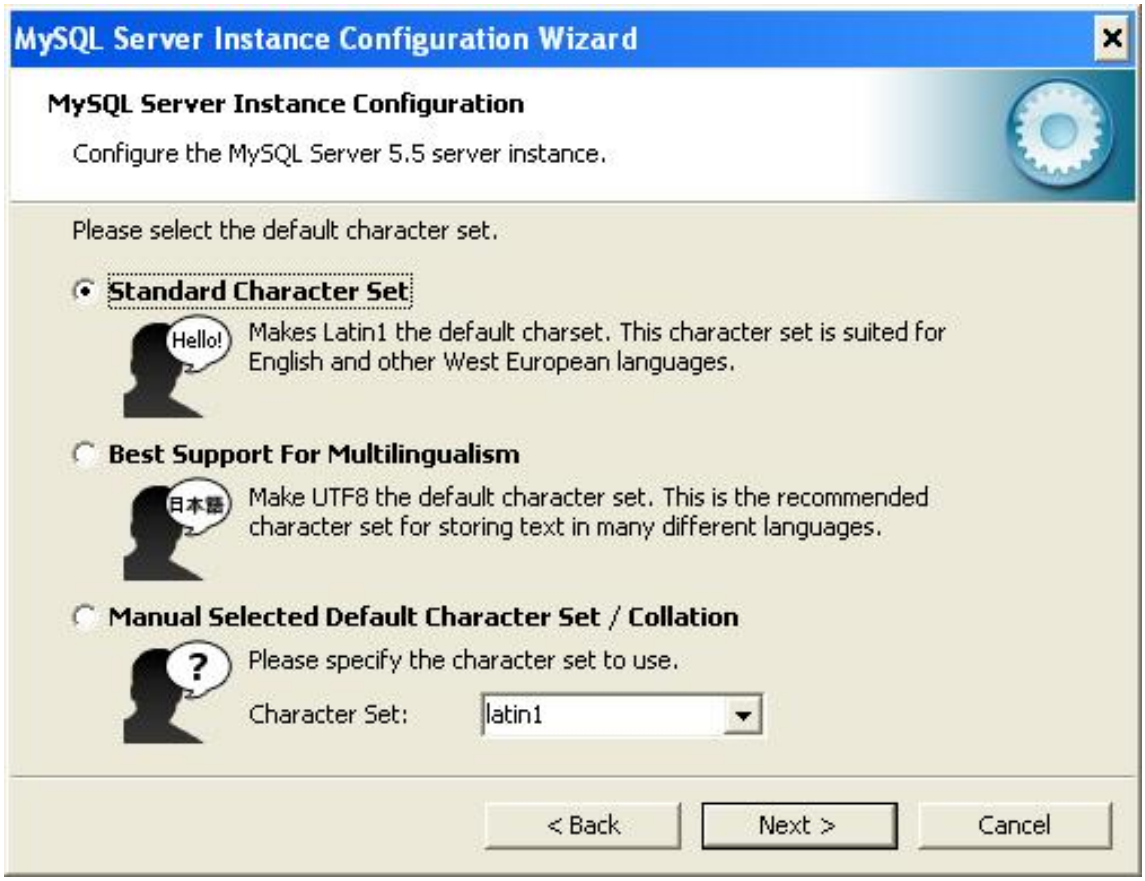
Bước 3: Xác định cổng làm việc của MySQL Server

- Với việc lựa chọn TCP/IP cho phép các máy kết nối theo giao thức TCP/IP; ngược lại, chỉ cho phép các kết nối cục bộ. Khi đã lựa chọn TCP/IP, chúng ta phải xác định ***Port Number***: số hiệu cổng làm việc của MySQL server. Cổng ngầm định MySQL là 3306.
- ***Enable Strict Mode***: nếu tùy chọn này được sử dụng, sẽ không cho phép đưa các giá trị không hợp lệ vào bảng dữ liệu: ví dụ dữ liệu NULL vào cột NOT NULL.



Bước 4: Lựa chọn hệ mã ký tự sử dụng khi lưu trữ

- Standard Character Set: ngầm định sử dụng tập chữ latin (ANSI)
- Best Support for Multilingualism: Với lựa chọn này, Unicode UTF8 được ngầm định sử dụng (*thích hợp với Việt Nam*).
- Manual Selected Default Character Set/Collation: cho phép lựa chọn hệ kí tự cụ thể khác trong hộp Character set.



Bước 5: Cấu hình tài khoản quản trị MySQL server

Bước này thiết lập mật khẩu cho tài khoản *root* quản trị hệ thống.

- Nếu ***Enable root access from remote machines*** được chọn. Tài khoản này có thể đăng nhập quản trị MySQL từ máy tính ở xa.
- ***Anonymous Account***: nếu được lựa chọn, thì người dùng bất kỳ có thể đăng nhập vào hệ thống (chỉ nên sử dụng trong quá trình phát triển, kiểm thử, không sử dụng khi triển khai hệ thống).

2. Cấu trúc MyQSL Server

File cấu hình

Tất cả các cấu hình cài đặt hệ thống đều được lưu lại trong file cấu hình. Tên file là *my.ini* nếu sử dụng Windows hoặc *my.cnf* Linux, Unix, và Mac. Nội dung chính của file cấu hình như sau (dòng bắt đầu bằng kí tự # là dòng chú thích):

```
# The TCP/IP Port the MySQL Server will listen on
```

```
port=3306
# Path to installation directory. All paths are
# usually resolved relative to this.
basedir="C:/Program Files/MySQL/MySQL Server 5.5/"
# Path to the database root
datadir="C:/Program Files/MySQL/MySQL Server 5.5/Data/"
```

- Tùy chọn *port:* xác định số hiệu cổng làm việc của MySQL Server
- Tùy chọn *basedir*: chỉ thư mục cài đặt MySQL server.
- Tùy chọn *datadir*: đường dẫn chỉ tới thư mục lưu trữ dữ liệu.

Gợi ý: Người sử dụng nên sử dụng thư mục làm việc và thư mục lưu trữ dữ liệu khác với cài đặt ngầm định để tăng tính bảo mật của hệ thống.

Cấu trúc thư mục MySQL

Thư mục	**Nội dung**
bin	File nhị phân - mysqld chương trình server, tất cả các chương trình khách và công cụ để sử dụng và quản trị MySQL server.
data	Nơi MySQL lưu trữ (đọc và ghi) dữ liệu, và các file log của server.
include	Tập các file header, sử dụng khi viết và biên dịch các chương trình sử dụng các thư viện của MySQL.
lib	Các file thư viện của MySQL.
scripts	mysql_install_db script, được sử dụng để khởi tạo file dữ liệu và các tài khoản.

share	SQL scripts để sửa các đặc quyền, cũng như tập các file ngôn ngữ.

- Thư mục **Bin** chứa các file chương trình của MySQL. Dưới đây là mô tả một số chương trình trong thư mục:

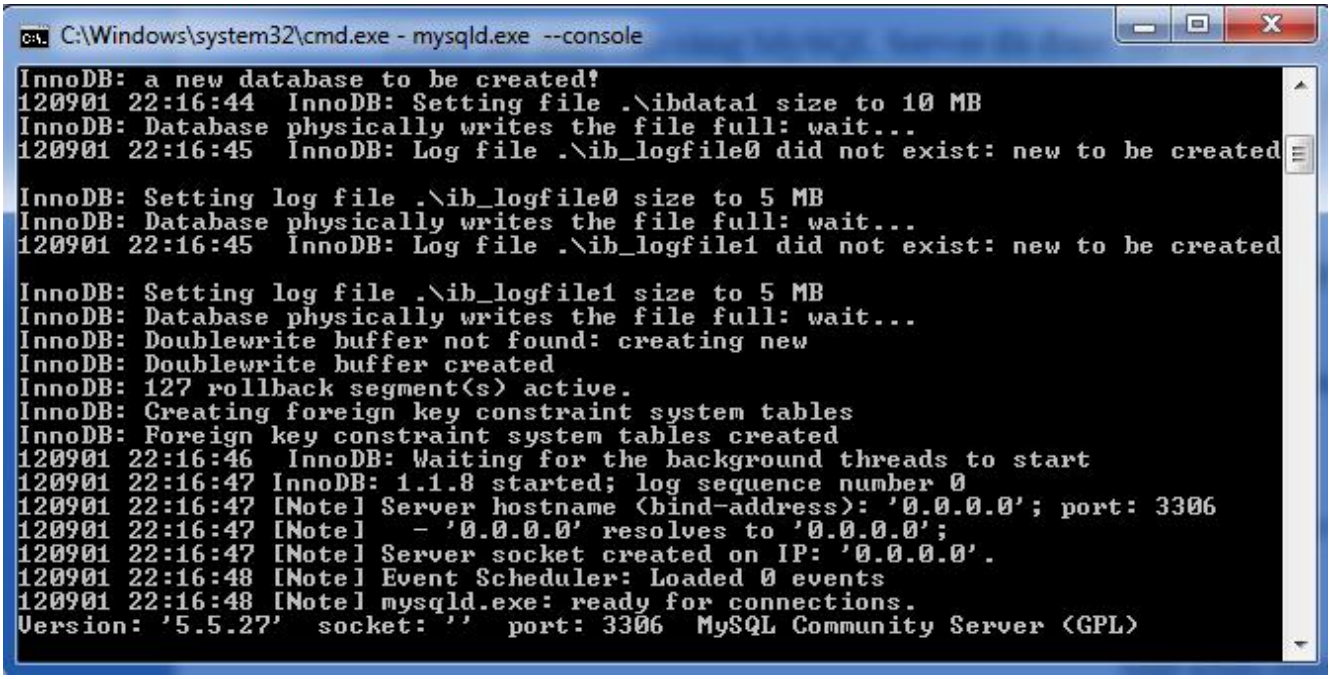
Tên chương trình	**Mô tả chức năng**
mysqld	MySQL server
mysql	Công cụ khách giúp thực thi tương tác các câu lệnh SQL
mysqladmin	Trợ giúp các tác vụ quản trị khác nhau (hiện thị trạng thái, tắt server,..).
mysqldump	Lưu nội dung của CSDL MySQL ra ngoài
mysqlimport	Nhập dữ liệu vào bảng từ file
mysqlshow	Hiển thị thông tin về CSDL, bảng, cột
myisamchk	Kiểm tra sự toàn vẹn của các file bảng MyISAM và sửa chữa
mysqlcheck	Thực hiện tác vụ bảo trì bảng

3. Kết nối tới MySQL server

Trước hết đảm bảo rằng MySQL Server đã được bật sau quá trình cài đặt trên.
Một cách khác có thể khởi động MySQL Server trực tiếp thông qua câu lệnh sau từ giao diên dòng lệnh command line.

```
basedir\mysqld.exe --console
```

Trong đó basedir là thư mục chứa chương trình mysqld.exe



Minh họa trên cho thấy tiến trình MySQL server đã chạy và chờ kết nối tới tại cổng có số hiệu 3306.

Chương trình khách khi kết nối tới MySQL server sử dụng một số tham số như trong bảng dưới, hai cách sử dụng là tương đương nhau.

-u <username>	--user=username	Xác định người dùng đăng nhập MySQL.
-p	--password	Hỏi mật khẩu ngay sau khi lệnh bắt đầu
-p<password>	--password=xxx	Mật khẩu được truyền trực tiếp. Khác với các lựa chọn khác, không có khoảng cách sau –p. Sẽ thuận tiện hơn nhưng giảm an toàn (nên tránh)
-h hostname	--host=hostname	Xác định tên hoặc địa chỉ IP của máy tính (giá trị ngầm định là chính máy tính *localhost*)
-P port	--port=port	Xác định cổng làm việc của MySQL server

Ví dụ: Hai cách đăng nhập vào hệ thống MySQL server sử dụng chương trình khách mysql.exe

Cách 1: Gõ lệnh sau từ cửa sổ lệnh

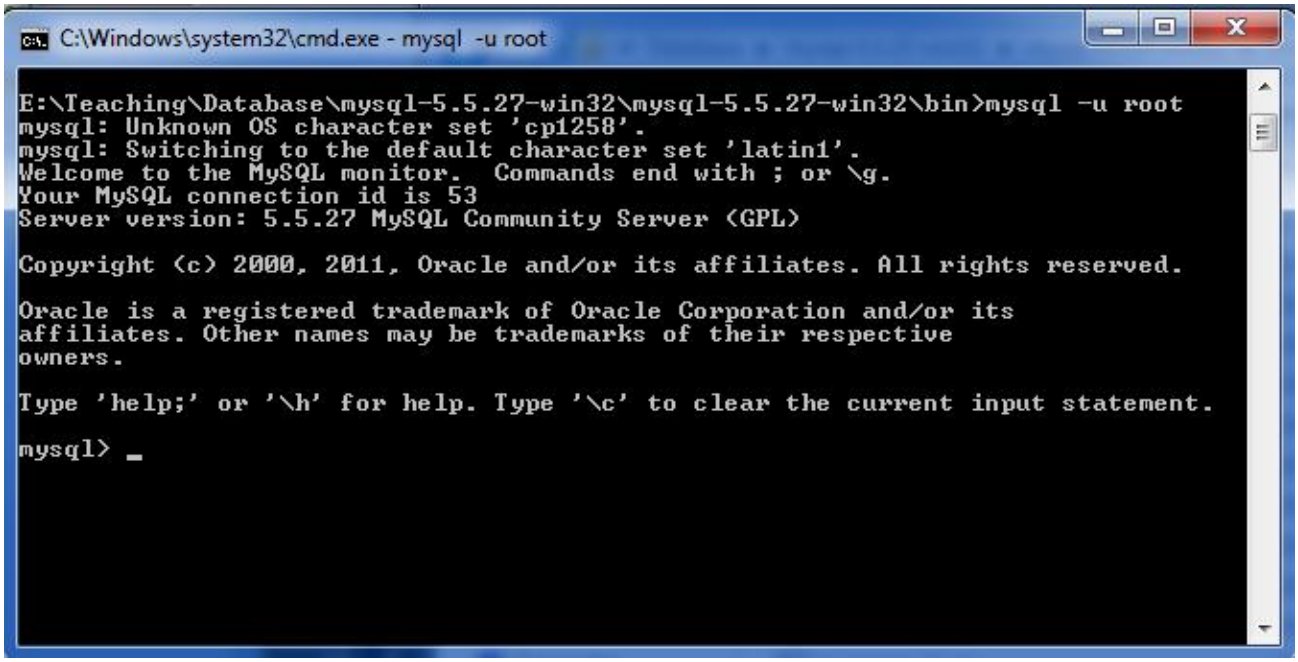
```
basedir\mysql.exe –u user_name –p your_password
```

Cách 2: Gõ lệnh sau từ cửa sổ lệnh

```
shell> basedir\mysql.exe --user=user_name --
password=your_password
```

Trong đó basedir là thư mục chứa chương trình mysqld.exe

Ngầm định ban đầu hệ quản trị CSDL có một tài khoản quản trị username là root và mật khẩu để trống.



* Bên cạnh sử dụng chương trình khách mysql.exe để kết nối làm việc với mysql server, bạn có thể sử dụng chương trình khách khác như mysql workbench, php myadmin..Sau khi kết nối thành công tới MySQL Server như hình trên, ta có thể thao tác với CSDL, V**í dụ**: `mysql> show databases;`

Ngắt kết nối tới MySQL server sử dụng:

```
mysql> exit;
```

4. Tạo, xóa cơ sở dữ liệu (CSDL)

- *Sau khi đã đăng nhập vào MySQL server sử dụng chương trình khách mysql.exe*, các bước sau mô tả cách khởi tạo và xóa cơ sở dữ liệu.**Khởi tạo CSDL**

Để tạo CSDL trong MySQL, sử dụng câu lệnh CREATE DATABASE như sau:

```
CREATE DATABASE [IF NOT EXISTS] database_name;
```

Chú ý: Các câu lệnh SQL kết thúc bởi dấu ; hoặc \g, \G và bấm phím Enter.

Câu lệnh CREATE DATABASE sẽ tạo CSDL có tên là *database_name*được xác định. IF NOT EXISTS là một tùy chọn tránh lỗi nếu tồn tại một CSDL cùng tên. Nếu đã tồn tại CSDL cùng tên trong MySQL server, câu lệnh sẽ không được thi hành.

Ví dụ: tạo một CSDL tên là *classicmodels*

```
CREATE DATABASE classicmodels;
```

- **Hiển thị các CSDL**

Câu lệnh SHOW DATABASES sẽ hiển thị tất cả các CSDL trong server. Có thể sử dụng câu lệnh này để kiểm tra CSDL mới tạo hoặc hiển thị tên tất cả các CSDL đã có trong server trước khi tạo CSDL mới.

```
SHOW DATABASES;
```

Database
information_schema
classicmodels
ebookshop
mysql
performance_schema
sakila
test

- **Chọn CSDL để làm việc**

Để chọn một CSDL có dự định làm việc, có thể sử dụng câu lệnh USE như sau:

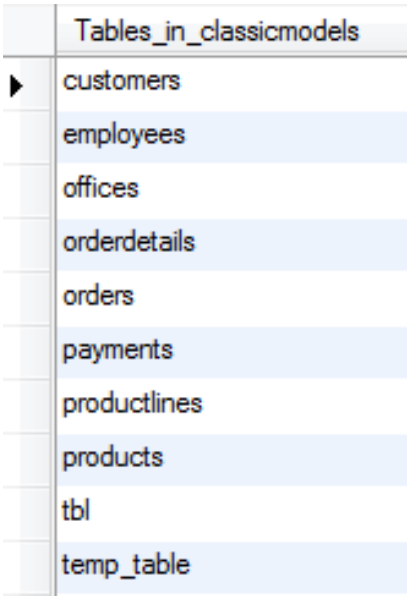
```
USE database_name;
```

Ví dụ: chọn CSDL classicmodels, sử dụng câu lệnh sau

```
USE classicmodels;
```

Từ đây có thể thao tác trên các bảng dữ liệu của CSDL được chọn. Ví dụ để hiển thị các bảng dữ liệu trong CSDL hiện thời sử dụng lệnh:

```
SHOW TABLES
```



Tables_in_classicmodels
customers
employees
offices
orderdetails
orders
payments
productlines
products
tbl
temp_table

- **Xóa Cơ sở Dữ liệu**

Xóa CSDL có nghĩa là sẽ xóa CSDL vật lý, tất cả dữ liệu và các đối tượng liên quan trong CSDL sẽ bị xóa vĩnh viễn. Do đó cần cẩn thận khi thi hành câu lệnh này.
MySQL cung cấp câu lệnh theo chuẩn DROP DATABASE để cho phép xóa một CSDL

```
DROP DATABASE [IF EXISTS] database_name;
```

Giống như câu lệnh CREATE DATABASE, tùy chọn IF EXIST chống xóa CSDL nếu không tồn tại.

❖ **Bài tập thực hành:**

1. Thay đổi cổng ngầm định của MySQL server thành 3307 và kết nối tới MySQL server tại cổng này.
2. Thay đổi đường dẫn ngầm định thư mục chứa CSDL trong file cấu hình
3. Tạo CDSL tên là *my_database*, sau đó dùng lệnh hiện thị các CSDL có trong server.
4. Kiểm tra trong thư mục chứa CSDL xem CSDL mới được tạo ra.
5. Xóa CSDL *my_database,* sau đó dùng lệnh hiển thị các CSDL có trong server.

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:

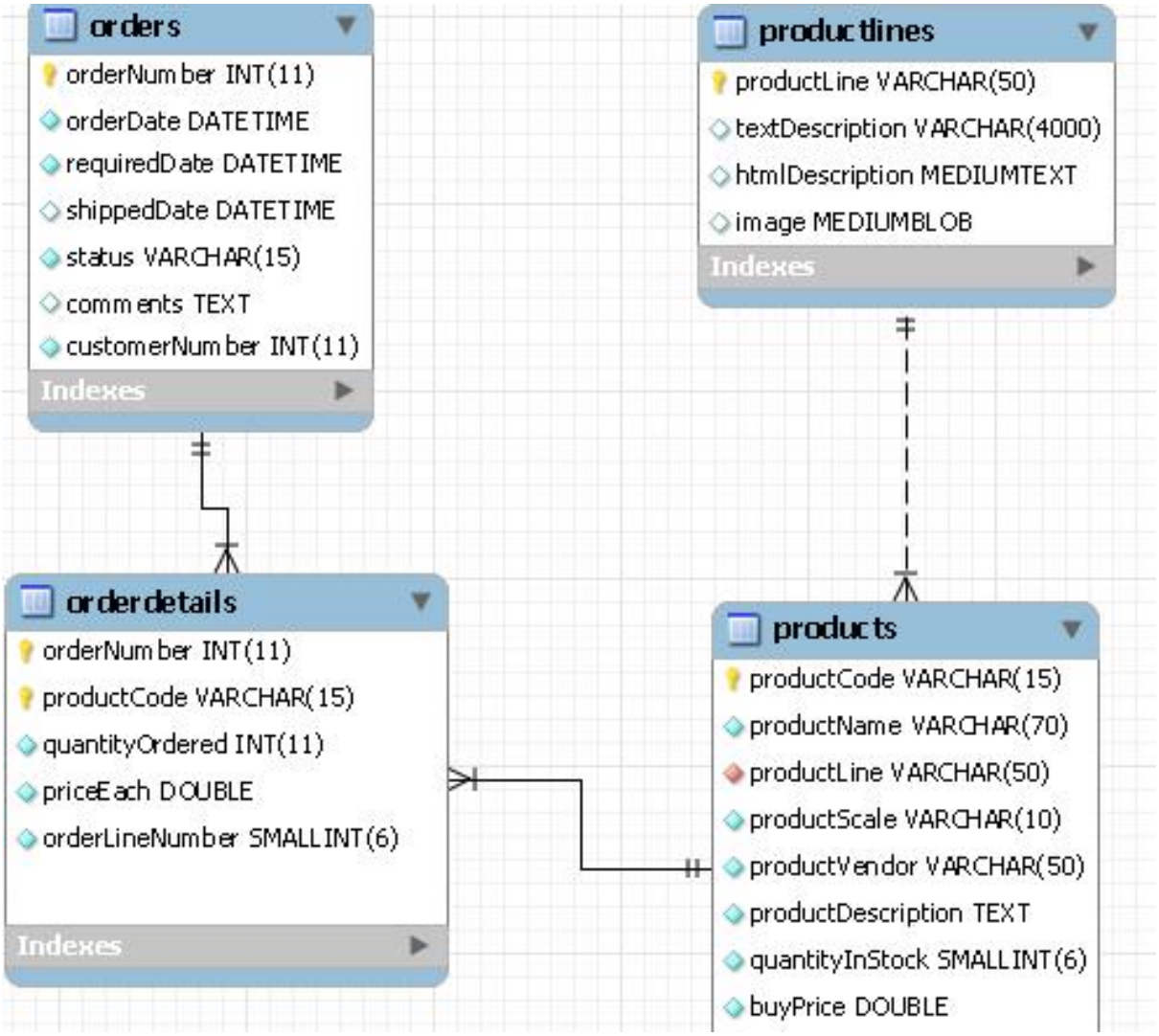
```
DROP TABLE [IF EXISTS] <table_name>
```

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

Bài thực hành số 3

Truy vấn cơ bản (phần 1)

❖ Nội dung chính

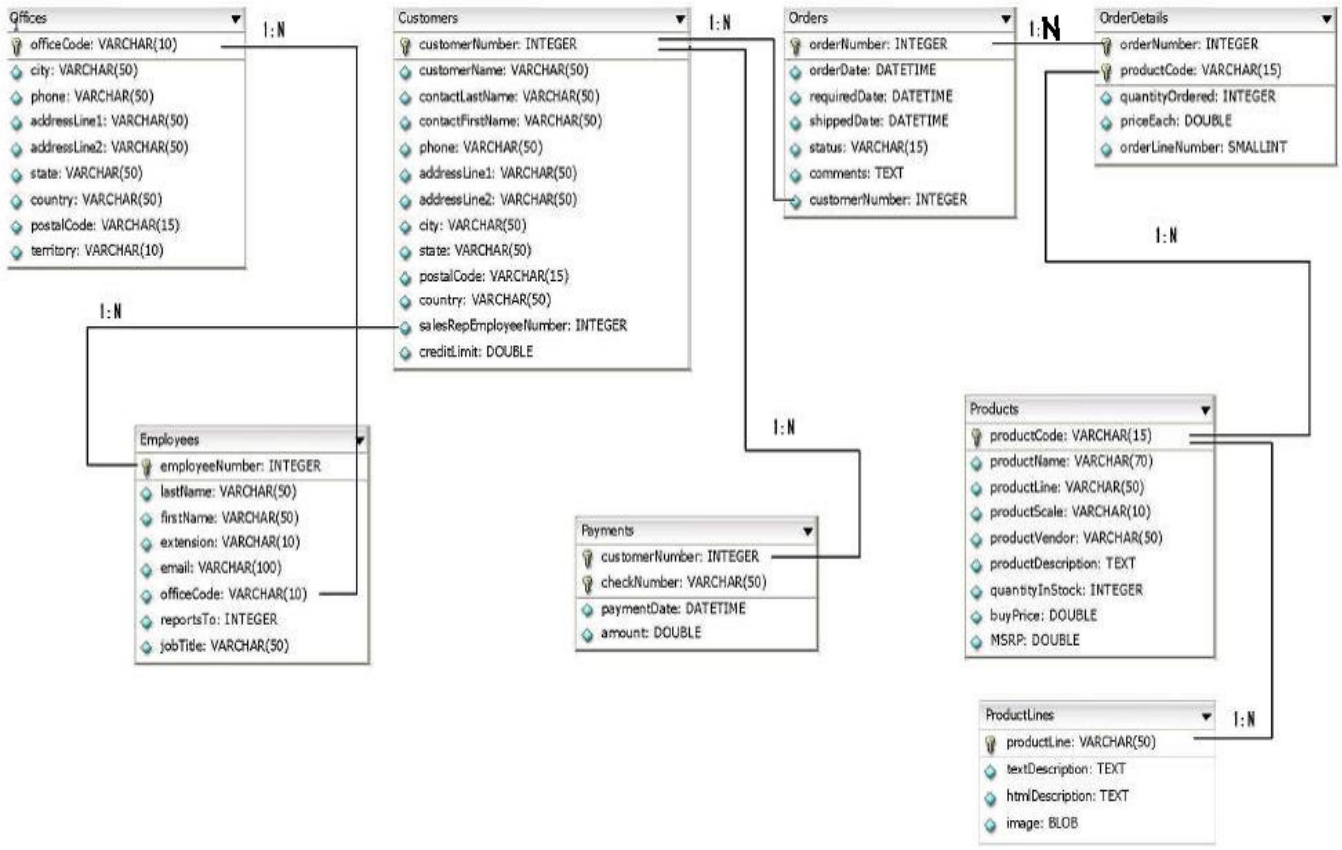
- Câu lệnh Select: cú pháp và cách sử dụng
- Mệnh đề where
- Loại bỏ dữ liệu kết quả trùng lặp với DISTINCT
- Giới hạn các bản ghi trả về bằng LIMIT

1. Cài đặt cở sở dữ liệu mẫu

Cơ sở dữ liệu mẫu bao gồm các bảng sau:

- *Customers*: Lưu trữ thông tin về khách hàng.
- *Products*: Lưu trữ danh sách về các sản phẩm.
- *ProductLines*: Lưu trữ danh mục các loại sản phẩm
- *Orders*: Lưu trữ các đơn hàng được đặt bởi các khách hàng.
- *OrderDetails*: Lưu trữ về chi tiết các dòng đơn hàng
- *Payments*: Lưu trữ các thanh toán của khách hàng
- *Employees*: Lưu trữ thông tin về các nhân viên của tổ chức
- *Offices*: Lưu thông tin về các văn phòng của tổ chức.

Hình dưới minh họa mối quan hệ giữa các bảng dữ liệu trong cơ sở dữ liệu



Tải file script *sampledatabase.sql* để tạo CSDLvề từ địa chỉ:
http://www.mysqltutorial.org/mysql-sample-database.aspx

Giả sử file *sampledatabase.sql* được đặt trong thư mục gốc ổ C:

Đăng nhập vào MySQL server từ chương trình khách *mysql.exe* sử dụng tài khoản root

Từ dấu nhắc mysql> thi hành câu lệnh sau:

```
source c:\sampledatabase.sql
```

Cơ sở dữ liệu được tạo ra có tên là *classicmodels*

2. Thực hiện truy vấn với câu lệnh SELECT

Trong phần này, sẽ học cách sử dụng mệnh đề SELECT để truy vấn dữ liệu từ các bảng cơ sở dữ liệu.

Cú pháp SELECT

```
SELECT tên cột 1, tên cột 2, ...
FROM các bảng
[WHERE điều kiện chọn]
[GROUP BY nhóm]
[HAVING điều kiện chọn nhóm]
[ORDER BY các cột sắp xếp]
[LIMIT giới hạn số lượng];
```

- Trong một truy vấn SELECT có nhiều yếu tố tùy chọn mà có thể sử dụng. Các tùy chọn được đặt trong dấu ngoặc vuông [].
- Thứ tự xuất hiện của các từ khoá WHERE, GROUP BY, HAVING, ORDER BY và LIMIT phải theo đúng thứ tự trên.

Để chọn tất cả các cột trong một bảng có thể sử dụng dấu sao (*) ký hiệu thay vì liệt kê tất cả các tên cột sau từ khoá SELECT.

Ví dụ: nếu cần phải truy vấn tất cả các thông tin về nhân viên, có thể sử dụng truy vấn sau đây:

```
SELECT * FROM employees
```

employeeNumber	lastName	firstName	extension	email	officeCode	reportsTo	jobTitle
1002	Murphy	Diane	x5800	dmurphy@classicmodelcars.com	1	NULL	President
1056	Patterson	Mary	x4611	mpatterso@classicmodelcars.com	1	1002	VP Sales
1076	Firrelli	Jeff	x9273	jfirrelli@classicmodelcars.com	1	1002	VP Marketing
1088	Patterson	William	x4871	wpatterson@classicmodelcars.com	6	1056	Sales Manager (APAC)
1102	Bondur	Gerard	x5408	gbondur@classicmodelcars.com	4	1056	Sale Manager (EMEA)
1143	Bow	Anthony	x5428	abow@classicmodelcars.com	1	1056	Sales Manager (NA)
1165	Jennings	Leslie	x3291	ljennings@classicmodelcars.com	1	1143	Sales Rep
1166	Thompson	Leslie	x4065	lthompson@classicmodelcars.com	1	1143	Sales Rep
1188	Firrelli	Julie	x2173	jfirrelli@classicmodelcars.com	2	1143	Sales Rep
1216	Patterson	Steve	x4334	spatterson@classicmodelcars.com	2	1143	Sales Rep
1286	Tseng	Foon Yue	x2248	ftseng@classicmodelcars.com	3	1143	Sales Rep
1323	Vanauf	George	x4102	gvanauf@classicmodelcars.com	3	1143	Sales Rep
1337	Bondur	Loui	x6493	lbondur@classicmodelcars.com	4	1102	Sales Rep
1370	Hernandez	Gerard	x2028	ghernande@classicmodelcars.com	4	1102	Sales Rep

cũng có thể xem dữ liệu một phần của một bảng bằng cách liệt kê tên các cột sau từ khóa SELECT. Điều này được gọi là *phépchiếu*.

Ví dụ: nếu cần phải xem *tên, họ và vị trí công việc* của nhân viên, có thể sử dụng truy vấn sau đây:

```
SELECT lastname, firstname, jobtitle
FROM Employees
```

lastname	firstname	jobtitle
Murphy	Diane	President
Patterson	Mary	VP Sales
Firrelli	Jeff	VP Marketing
Patterson	William	Sales Manager (APAC)
Bondur	Gerard	Sale Manager (EMEA)
Bow	Anthony	Sales Manager (NA)
Jennings	Leslie	Sales Rep
Thompson	Leslie	Sales Rep
Firrelli	Julie	Sales Rep
Patterson	Steve	Sales Rep
Tseng	Foon Yue	Sales Rep
Vanauf	George	Sales Rep
Bondur	Loui	Sales Rep
Hernandez	Gerard	Sales Rep
Castillo	Pamela	Sales Rep

Ví dụ: Muốn lấy ra thông tin về *mã sản phẩm* và *tên sản phẩm*, thực hiện truy vấn như sau:

```
SELECT ProductCode, ProductName
FROM Products
```

ProductCode	ProductName
S10_1678	1969 Harley Davidson Ultimate Chopper
S10_1949	1952 Alpine Renault 1300
S10_2016	1996 Moto Guzzi 1100i
S10_4698	2003 Harley-Davidson Eagle Drag Bike
S10_4757	1972 Alfa Romeo GTA
S10_4962	1962 LanciaA Delta 16V
S12_1099	1968 Ford Mustang
S12_1108	2001 Ferrari Enzo
S12_1666	1958 Setra Bus
S12_2823	2002 Suzuki XREO
S12_3148	1969 Corvair Monza
S12_3380	1968 Dodge Charger
S12_3891	1969 Ford Falcon
S12_3990	1970 Plymouth Hemi Cuda
S12_4473	1957 Chevy Pickup

3. Mệnh đề WHERE

Mệnh đề WHERE của câu lệnh SELECT cho phép chọn các hàng cụ thể phù hợp với điều kiện hoặc tiêu chí tìm kiếm. Sử dụng mệnh đề WHERE để lọc các bản ghi dựa trên một điều kiện nhất định.

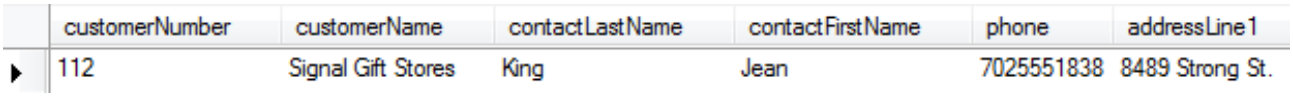
Ví dụ: có thể tìm thấy các chủ tịch của công ty bằng cách sử dụng truy vấn sau đây:

```
SELECT FirstName, LastName, email
FROM Employees
WHERE jobtitle = "President"
```

firstname	lastname	email
Diane	Murphy	dmurphy@classicmodelcars.com

Hoặc có thể tìm ra các thông tin về tên của khách hàng có mã số 112 bằng truy vấn như sau:

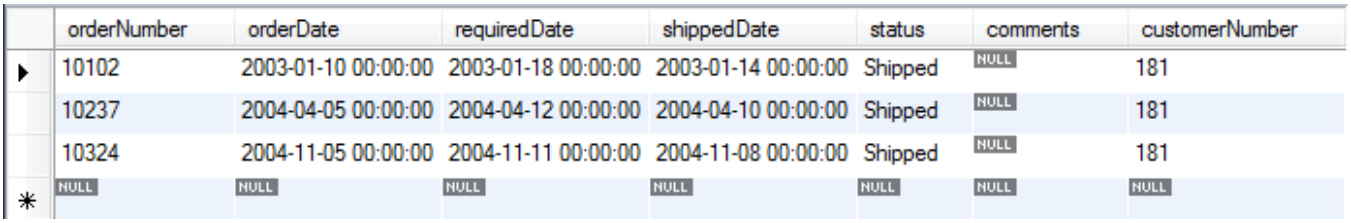
```
SELECT *
FROM Customers
WHERE customerNumber=112
```



customerNumber	customerName	contactLastName	contactFirstName	phone	addressLine1
112	Signal Gift Stores	King	Jean	7025551838	8489 Strong St.

Ví dụ sau đưa ra các đơn hàng có mã khách hàng là 181

```
SELECT *
FROM orders
WHERE customerNumber = 181
```



orderNumber	orderDate	requiredDate	shippedDate	status	comments	customerNumber
10102	2003-01-10 00:00:00	2003-01-18 00:00:00	2003-01-14 00:00:00	Shipped	NULL	181
10237	2004-04-05 00:00:00	2004-04-12 00:00:00	2004-04-10 00:00:00	Shipped	NULL	181
10324	2004-11-05 00:00:00	2004-11-11 00:00:00	2004-11-08 00:00:00	Shipped	NULL	181
NULL	NULL	NULL	NULL	NULL	NULL	NULL

4. Kết nối các điều kiện với toán tử AND và OR

Chúng ta có thể kết hợp hai hay nhiều điều kiện khác nhau trong mệnh đề WHERE, sử dụng các toán tử **AND, OR.** Với hai điều kiện nối bởi AND, cần cả hai đúng để điều kiện kết hợp là đúng. Với hai điều kiện nối bởi OR, điều kiện kết hợp là đúng nếu một hoặc cả hai điều kiện là đúng

Ví dụ: đưa ra các khách hàng tại Mỹ của người chăm sóc khách hàng có mã là 1165

```
SELECT *
FROM customers
WHERE country ='USA' and salesRepEmployeeNumber = 1165
```

contactFirstName	phone	addressLine1	addressLine2	city	state	postalCode	country	salesRepEmployeeNumber
Susan	4155551450	5677 Strong St.	NULL	San Rafael	CA	97562	USA	1165
Julie	6505555787	5557 North Pendale Street	NULL	San Francisco	CA	94217	USA	1165
Juri	6505556809	9408 Furth Circle	NULL	Burlingame	CA	94217	USA	1165
Julie	6505551386	7734 Strong St.	NULL	San Francisco	CA	94217	USA	1165
Sue	4085553659	3086 Ingle Ln.	NULL	San Jose	CA	94217	USA	1165
Sue	4155554312	2793 Furth Circle	NULL	Brisbane	CA	94217	USA	1165

Ví dụ: đưa ra các đơn hàng có trạng thái là ‘On Hold’ hoặc ‘In Process’

```
SELECT *
FROM orders
WHERE status = 'On Hold' or status ='In Process'
```

orderNumber	orderDate	requiredDate	shippedDate	status	comments
10334	2004-11-19 00:00:00	2004-11-28 00:00:00	NULL	On Hold	The outstaniding balance for this customer exceeds their credit limit. Order will
10401	2005-04-03 00:00:00	2005-04-14 00:00:00	NULL	On Hold	Customer credit limit exceeded. Will ship when a payment is received.
10407	2005-04-22 00:00:00	2005-05-04 00:00:00	NULL	On Hold	Customer credit limit exceeded. Will ship when a payment is received.
10414	2005-05-06 00:00:00	2005-05-13 00:00:00	NULL	On Hold	Customer credit limit exceeded. Will ship when a payment is received.
10420	2005-05-29 00:00:00	2005-06-07 00:00:00	NULL	In Process	NULL
10421	2005-05-29 00:00:00	2005-06-06 00:00:00	NULL	In Process	Custom shipping instructions were sent to warehouse
10422	2005-05-30 00:00:00	2005-06-11 00:00:00	NULL	In Process	NULL
10423	2005-05-30 00:00:00	2005-06-05 00:00:00	NULL	In Process	NULL
10424	2005-05-31 00:00:00	2005-06-08 00:00:00	NULL	In Process	NULL
10425	2005-05-31 00:00:00	2005-06-07 00:00:00	NULL	In Process	NULL

5. IS NULL: tìm các giá trị không xác định

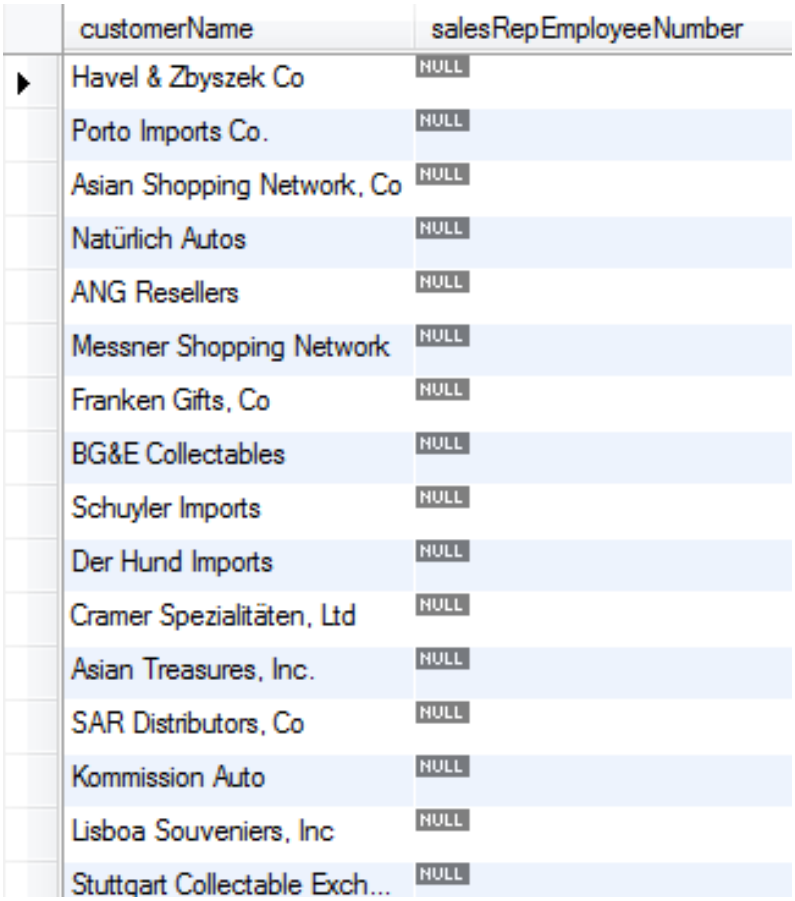
Với các trường chưa được nhập dữ liệu (coi giá trị là chưa xác định), SQL coi giá trị đó là NULL. Để kiểm tra một trường có giá trị là NULL hay không, thay vì sử dụng phép so sánh =, SQL sử dụng phép toán *is NULL*

Ví dụ: Đưa ra các khách hàng chưa được gán nhân viên chăm sóc

```
SELECT customerName, salesRepEmployeeNumber

FROM customers

WHERE salesRepEmployeeNumber = NULL
```

Nếu sử dụng phép so sánh = như trên, sẽ không có dòng kết quả nào được trả về. Nếu thay phép so sánh = bởi *is NULL*

```
SELECT customerName, salesRepEmployeeNumber

FROM customers

WHERE salesRepEmployeeNumber is NULL
```



customerName	salesRepEmployeeNumber
Havel & Zbyszek Co	NULL
Porto Imports Co.	NULL
Asian Shopping Network, Co	NULL
Natürlich Autos	NULL
ANG Resellers	NULL
Messner Shopping Network	NULL
Franken Gifts, Co	NULL
BG&E Collectables	NULL
Schuyler Imports	NULL
Der Hund Imports	NULL
Cramer Spezialitäten, Ltd	NULL
Asian Treasures, Inc.	NULL
SAR Distributors, Co	NULL
Kommission Auto	NULL
Lisboa Souveniers, Inc	NULL
Stuttgart Collectable Exch...	NULL

6. Từ khoá DISTINCT

Với từ khóa DISTINCT, có thể loại bỏ dữ liệu trùng lặp từ câu lệnh SELECT.

Ví dụ: để tìm thấy bao nhiêu *vị trí công việc* của tất cả các nhân viên, sử dụng từ khóa DISTINCT trong câu lệnh SELECT như sau:

```
SELECT DISTINCT jobTitle FROM Employees;
```

jobTitle
President
VP Sales
VP Marketing
Sales Manager (APAC)
Sale Manager (EMEA)
Sales Manager (NA)
Sales Rep

Hoặc có thể tìm ra mã số các sản phẩm đã được mua bằng truy vấn như sau:

```
SELECT DISTINCT productCode FROM OrderDetails;
```

productCode
S18_1749
S18_2248
S18_4409
S24_3969
S18_2325
S18_2795
S24_1937
S24_2022
S18_1342
S18_1367
S10_1949
S10_4962
S12_1666
S18_1097
S18_2432

7. Giới hạn số lượng kết quả với LIMIT

Trong hầu hết các lần truy vấn, khi làm việc với các bảng dữ liệu có chứa hàng nghìn đến hàng triệu bản ghi và không muốn viết một truy vấn để có được tất cả các dữ liệu đó để đảm bảo hiệu suất và lưu lượng truy cập giữa các máy chủ cơ sở dữ liệu và máy chủ ứng

dụng . MySQL hỗ trợ một tính năng là LIMIT cho phép hạn chế các bản ghi trả lại với câu lệnh SELECT.

Giả thiết ta có một bảng cơ sở dữ liệu với 10.000 bản ghi và muốn nhận được N bản ghi đầu tiên, có thể sử dụng truy vấn sau đây:

```
SELECT * FROM table_name
LIMIT N
```

LIMIT cũng cho phép lấy ra một số lượng bản ghi nhất định tính từ một vị trí nào đó:

```
LIMIT S, N
```

Trong câu truy vấn trên, S là điểm bắt đầu ghi chỉ số. MySQL xác định rằng vị trí đầu tiên được ghi lại bắt đầu với 0; N là số lượng bản ghi muốn chọn.

Ví dụ: Có thể lấy ra thông tin về tên của 5 sản phẩm đầu tiên trong bảng Product bằng truy vấn như sau:

```
SELECT productName FROM Products
LIMIT 5;
```

productName
1969 Harley Davidson Ultimate Chopper
1952 Alpine Renault 1300
1996 Moto Guzzi 1100i
2003 Harley-Davidson Eagle Drag Bike
1972 Alfa Romeo GTA

Hoặc có thể lấy ra thông tin về 10 khách hàng đầu tiên hiện đang ở Pháp bằng truy vấn như sau:

```
select * from customers
where country='France'
limit 10;
```

customerNumber	customerName	contactLastName	contactFirstName	phone	addressLine1	address
103	Atelier graphique	Schmitt	Carine	40.32.2555	54, rue Royale	NULL
119	La Rochelle Gifts	Labrune	Janine	40.67.8555	67, rue des Cinquante Otages	NULL
146	Saveley & Henriot, Co.	Saveley	Mary	78.32.5555	2, rue du Commerce	NULL
171	Daedalus Designs Imports	Rancé	Martine	20.16.1555	184, chaussée de Tournai	NULL
172	La Corne D'abondance, Co.	Bertrand	Marie	(1) 42.34.2555	265, boulevard Charonne	NULL
209	Mini Caravy	Citeaux	Frédérique	88.60.1555	24, place Kléber	NULL
242	Alpha Cognac	Roulet	Annette	61.77.6555	1 rue Alsace-Lorraine	NULL
250	Lyon Souveniers	Da Silva	Daniel	+33 1 46 62 7555	27 rue du Colonel Pierre Avia	NULL
256	Auto Associés & Cie.	Tonini	Daniel	30.59.8555	67, avenue de l'Europe	NULL
350	Marseille Mini Autos	Lebihan	Laurence	91.24.4555	12, rue des Bouchers	NULL

❖ Bài tập thực hành:

1. Đưa ra danh sách các nhân viên có trường reportsTo chưa xác định.
2. Đưa ra danh sách các CustomerNumber đã có thực hiện giao dịch.
3. Đưa ra danh sách các đơn hàng có ngày yêu cầu vận chuyển là ‘18/1/2003’. *Lưu ý*: MySQL lưu dữ liệu ngày tháng theo định dạng năm/tháng/ngày.
4. Đưa ra danh sách các đơn hàng có ngày đặt trong tháng 4 năm 2005 và có trạng thái là ‘Shipped’
5. Đưa ra danh sách các sản phẩm thuộc nhóm ‘Classic Cars’.

Bài thực hành số 4

Truy vấn cơ bản (phần 2)

❖ Nội dung chính

Trong bài này, sẽ đề cập đến cách sử dụng một số toán tử như *IN, BETWEEN, UNION, LIKE, ORDER BY; Thuộc tính suy diễn.*

1. Toán tử IN

Toán tử IN cho phép chọn giá trị phù hợp từ một tập các giá trị. Cú pháp sử dụng như sau:

```
SELECT danh sách các cột
FROM tên bảng
WHERE cột IN ("giá trị 1","giá trị 2"…)
```

Các cột trong mệnh đề WHERE không cần phải xuất hiện trong danh sách cột đã chọn, nhưng nó phải là một cột trong bảng. Nếu danh sách có nhiều hơn một giá trị, mỗi mục được phân cách bằng dấu phẩy. Ngoài ra, có thể sử dụng toán tử NOT đi kèm với toán tử IN cho mục đích phủ định.

Chúng ta hãy xem một số ví dụ sau:

Giả sử nếu muốn tìm tất cả các văn phòng được đặt tại Mỹ (USA) và Pháp (France), có thể thực hiện truy vấn sau đây:

```
SELECT officeCode, city, phone
FROM offices
WHERE country = 'USA' OR country = 'France'
```

Trong trường hợp này, chúng ta có thể sử dụng IN thay vì truy vấn trên:

```
SELECT officeCode, city, phone
FROM offices
WHERE country IN ('USA','France')
```

Kết quả trả về như sau:

officeCode	city	phone
1	San Francisco	+1 650 219 4782
2	Boston	+1 215 837 0825
3	NYC	+1 212 555 3000
4	Paris	+33 14 723 4404

Để có được tất cả các văn phòng không nằm ở Mỹ và Pháp, chúng ta có thể sử dụng NOT IN như sau:

```
SELECT officeCode, city, phone
FROM offices
WHERE country NOT IN ('USA','France')
```

Kết quả trả về như sau:
Kết quả trả về như sau:

officeCode	city	phone
5	Tokyo	+81 33 224 5000
6	Sydney	+61 2 9264 2451
7	London	+44 20 7877 2041

2. Toán tử BETWEEN

BETWEEN cho phép lấy các giá trị trong một phạm vi cụ thể. Nó phải được sử dụng trong mệnh đề WHERE. Sau đây minh họa cú pháp:

```
SELECT column_list
FROM table_name
WHERE column_1 BETWEEN lower_range AND upper_range
```

MySQL trả lại tất cả bản ghi trong đó giá trị column_1 nằm trong phạm vi lower_rage và upper_range. Truy vấn tương đương để có được cùng một kết quả là:

```
SELECT column_list
FROM table_name
WHERE column_1 >= lower_range AND column_1 <= upper_range
```

Ví dụ:

Giả sử chúng ta muốn tìm tất cả các sản phẩm có giá nằm trong phạm vi 90 $ và 100 $, chúng ta có thể thực hiện truy vấn sau đây:

```
SELECT productCode,ProductName,buyPrice
FROM products
WHERE buyPrice BETWEEN 90 AND 100
ORDER BY buyPrice DESC
```

Kết quả trả về như sau:

productCode	ProductName	buyPrice
S10_1949	1952 Alpine Renault 1300	98.58
S24_3856	1956 Porsche 356A Coupe	98.3
S12_1108	2001 Ferrari Enzo	95.59
S12_1099	1968 Ford Mustang	95.34
S18_1984	1995 Honda Civic	93.89
S18_4027	1970 Triumph Spitfire	91.92
S10_4698	2003 Harley-Davidson Eagle Drag Bike	91.02

Để tìm tất cả các bản ghi không nằm trong một phạm vi, chúng ta sử dụng NOT BETWEEN. Ví dụ: để tìm tất cả các sản phẩm với giá mua nằm ngoài phạm vi 20 và 100, chúng ta có thể viết truy vấn sau đây:

```
SELECT productCode,ProductName,buyPrice
FROM products
WHERE buyPrice NOT BETWEEN 20 AND 100
```

Kết quả trả về như sau:

productCode	ProductName	buyPrice
S10_4962	1962 LanciaA Delta 16V	103.42
S18_2238	1998 Chrysler Plymouth Prowler	101.51
S24_2840	1958 Chevy Corvette Limited Edition	15.91
S24_2972	1982 Lamborghini Diablo	16.24

Truy vấn trên tương đương với truy vấn sau:

```
SELECT productCode,ProductName,buyPrice
FROM products
WHERE buyPrice < 20 OR buyPrice > 100
ORDER BY buyPrice DESC
```

Kết quả trả về như sau:

productCode	ProductName	buyPrice
S10_4962	1962 LanciaA Delta 16V	103.42
S18_2238	1998 Chrysler Plymouth Prowler	101.51
S24_2972	1982 Lamborghini Diablo	16.24
S24_2840	1958 Chevy Corvette Limited Edition	15.91

3. Toán tử LIKE

LIKE cho phép thực hiện việc tìm kiếm thông tin dựa trên sự so sánh ký tự (‘*giống như’*). LIKE thường được sử dụng với câu lệnh SELECT trong mệnh đề WHERE. MySQL cung cấp cho hai ký tự đại diện sử dụng với LIKE, đó là % và _.

- Ký tự đại diện tỷ lệ phần trăm (%) đại diện cho bất kỳ chuỗi có thể không có hoặc có nhiều ký tự
- Gạch dưới (_) chỉ đại diện cho một ký tự duy nhất.

Ví dụ: Giả sử muốn tìm kiếm những nhân viên có tên bắt đầu với ký tự 'a', có thể làm điều đó như sau:

```
SELECT employeeNumber, lastName, firstName
 FROM employees
 WHERE firstName LIKE 'a%'
```

Kết quả trả về như sau:

employeeNumber	lastName	firstName
1143	Bow	Anthony
1611	Fixter	Andy

MySQL quét toàn bộ bảng employees (*nhân viên)* để tìm tất cả nhân viên có tên bắt đầu với ký tự 'a' và theo sau bởi một số lượng ký tự bất kỳ.

Ví dụ: Để tìm kiếm tất cả các nhân viên có họ kết thúc với chuỗi 'on', có thể thực hiện truy vấn như sau:

```
SELECT employeeNumber, lastName, firstName
 FROM employees
 WHERE lastName LIKE '%on'
```

Kết quả trả về như sau:

	employeeNumber	lastName	firstName
▶	1056	Patterson	Mary
	1088	Patterson	William
	1166	Thompson	Leslie
	1216	Patterson	Steve

Nếu chỉ biết rằng chuỗi tìm kiếm được nhúng vào một vị trí nào đó trong giá trị của một cột, có thể đặt % đầu và cuối của chuỗi tìm kiếm để tìm tất cả khả năng.

Ví dụ: muốn tìm tất cả các nhân viên mà họ của các nhân viên này có chứa cụm 'on', có thể thực hiện truy vấn sau đây:

```
SELECT employeeNumber, lastName, firstName
FROM employees
WHERE lastName LIKE '%on%'
```

Kết quả trả về như sau:

	employeeNumber	lastName	firstName
▶	1056	Patterson	Mary
	1088	Patterson	William
	1102	Bondur	Gerard
	1166	Thompson	Leslie
	1216	Patterson	Steve
	1337	Bondur	Loui
	1504	Jones	Barry

Chúng ta cũng có thể dùng NOT kèm với LIKE để hàm chứa ý nghĩa phủ định.
Ví dụ: muốn tìm các nhân viên có họ không bắt đầu bởi ký tự 'B', viết như sau:

```
SELECT employeeNumber, lastName, firstName
FROM employees
WHERE lastName NOT LIKE 'B%'
```

Kết quả trả về như sau:

employeeNumber	lastName	firstName
1002	Murphy	Diane
1056	Patterson	Mary
1076	Firrelli	Jeff
1088	Patterson	William
1165	Jennings	Leslie
1166	Thompson	Leslie
1188	Firrelli	Julie
1216	Patterson	Steve
1286	Tseng	Foon Yue
1323	Vanauf	George
1370	Hernandez	Gerard
1401	Castillo	Pamela
1504	Jones	Barry
1611	Fixter	Andy
1612	Marsh	Peter

Lưu ý là MySQL không phân biệt chữ hoa chữ thường nên 'b%' và 'B%' là như nhau.
Trong trường hợp chuỗi tìm kiếm của lại bắt đầu bởi một ký tự đại diện, chẳng hạn là '_', mysql cung cấp cho ký tự '\' để chỉ ra rằng các ký tự đại diện đi sau đó được sử dụng theo đúng nghĩa đen chứ không còn là ký tự đại diện nữa.
Ví dụ: tìm các sản phẩm mà mã của chúng có chứa chuỗi '_20', khi đó phải viết truy vấn như sau:

```
SELECT productCode, productName
FROM products
```

```
WHERE productCode LIKE '%\_20%'
```

Kết quả trả về như sau:

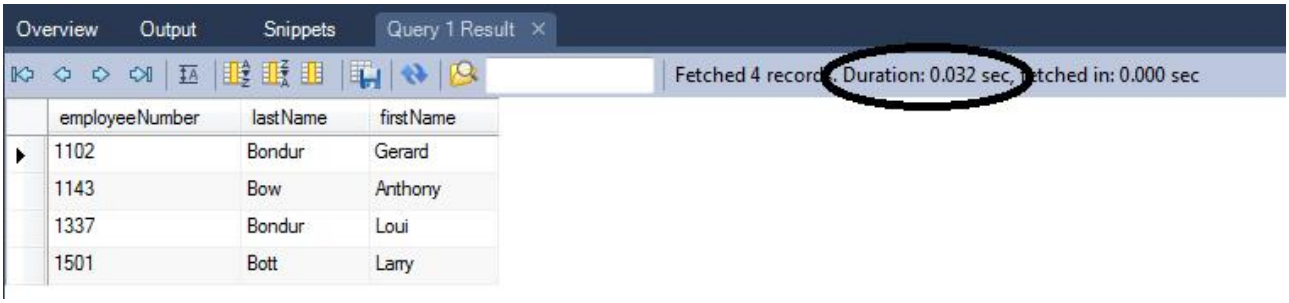
	productCode	productName
▶	S10_2016	1996 Moto Guzzi 1100i
	S24_2000	1960 BSA Gold Star DBD34
	S24_2011	18th century schooner
	S24_2022	1938 Cadillac V-16 Presidential Limousine
	S700_2047	HMS Bounty

LIKE cung cấp cho một cách thuận tiện để tìm bản ghi có các cột chứa các chuỗi phù hợp với mẫu tìm kiếm. Tuy nhiên, do việc thực thi LIKE chính là quét toàn bộ bảng để tìm tất cả các bản ghi phù hợp do đó nó không cho phép database engine sử dụng index để tìm kiếm nhanh. Khi dữ liệu trong bảng là đủ lớn, hiệu suất thực thi LIKE sẽ bị suy giảm. Trong một số trường hợp, có thể tránh vấn đề này bằng cách sử dụng các kỹ thuật khác để đạt được các kết quả tương tự. Hoặc sử dụng phương pháp đánh chỉ mục FULLTEXT của MySQL (sẽ đề cập về kỹ thuật này trong khóa học về Hệ quản trị CSDL MySQL).

Ví dụ: nếu muốn tìm tất cả các nhân viên có tên đầu tiên bắt đầu với một chuỗi quy định có thể sử dụng hàm LEFT() giống như các truy vấn sau đây:

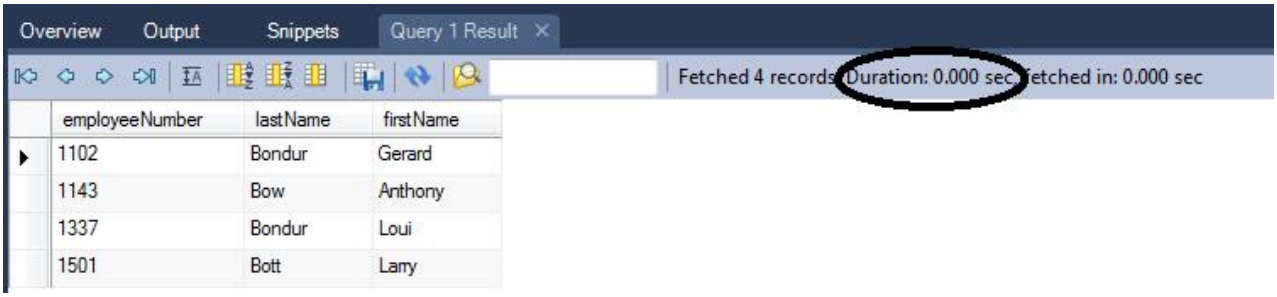
```
SET @str = 'b';
SELECT employeeNumber, lastName, firstName
FROM employees
WHERE LEFT(lastname,length(@str)) = @str;
```

Kết quả trả về như sau:



Kết quả trả về của truy vấn này là tương đương với truy vấn dưới đây, tuy nhiên tốc độ thực thi của cách viết sau tốt hơn rất nhiều vì chúng ta có thể sử dụng index trên cột *lastname*.

```
SELECT employeeNumber, lastName, firstName
FROM employees
WHERE lastname LIKE 'b%'
```



4. Thuộc tính suy diễn (Derived Attribute)

SQL cung cấp khả năng tạo các thuộc tính suy diễn trong bảng kết quả trả về sử dụng các toán tử và hàm dựa trên các thuộc tính có sẵn. Tên cột của thuộc tính suy diễn phụ thuộc vào hệ thống, tuy nhiên có thể gán bí danh làm tên cột.

Ví dụ sau sẽ tạo ra một cột suy diễn được đặt tên là lineTotal, thuộc tính này là kết quả phép nhân giữa hai thuộc tính `priceEach` và `quantityOrdered`

```
SELECT orderNumber, (priceEach*quantityOrdered) as
lineTotal
FROM orderdetails
```

orderNumber	lineTotal
10100	4080
10100	2754.5
10100	1660.12
10100	1729.21
10101	2701.5
10101	4343.56
10101	1463.8500000000001
10101	2040.1000000000001
10102	3726.45
10102	1768.3300000000002
10103	5571.8
10103	5026.14
10103	3284.28
10103	3307.5
10103	1283.48
10103	2488.12

5. Sắp xếp kết quả với ORDER BY

Mệnh đề ORDER BY cho phép sắp xếp các kết quả trên một hoặc nhiều cột trong kết quả truy vấn theo thứ tự tăng dần hay giảm dần. Để sắp xếp kết quả theo thứ tự tăng dần, sử dụng ASC; giảm dần là DESC. Theo mặc định, ORDER BY sẽ sắp xếp các kết quả theo thứ tự tăng dần.

Ví dụ: để sắp xếp danh sách nhân viên theo *tên* và *vị trí công việc*, có thể thực hiện truy vấn sau đây:

```
SELECT FirstName, LastName, jobtitle
FROM Employees
ORDER BY firstname ASC, jobtitle DESC;
```

FirstName	LastName	jobtitle
Andy	Fixter	Sales Rep
Anthony	Bow	Sales Manager (NA)
Barry	Jones	Sales Rep
Diane	Murphy	President
Foon Yue	Tseng	Sales Rep
George	Vanauf	Sales Rep
Gerard	Hernandez	Sales Rep
Gerard	Bondur	Sale Manager (EMEA)
Jeff	Firrelli	VP Marketing
Julie	Firrelli	Sales Rep
Larry	Bott	Sales Rep
Leslie	Jennings	Sales Rep
Leslie	Thompson	Sales Rep
Loui	Bondur	Sales Rep
Mami	Nishi	Sales Rep

Hoặc có thể đưa ra thông tin về tên các sản phẩm theo thứ tự tăng dần của số lượng hàng tồn kho bằng truy vấn như sau:

```
SELECT productName
FROM Products
ORDER BY quantityInStock;
```

Trong câu lệnh trên từ khóa ASC không sử dụng, do mặc định sẽ sắp xếp kết quả theo thứ tự tăng dần. Kết quả của câu lệnh trong hình sau.

productName
1960 BSA Gold Star DBD34
1968 Ford Mustang
1928 Ford Phaeton Deluxe
1997 BMW F650 ST
Pont Yacht
1911 Ford Town Car
1928 Mercedes-Benz SSK
F/A 18 Hornet 1/72
2002 Yamaha YZR M1
The Mayflower
1996 Peterbilt 379 Stake Bed with Outrigger
P-51-D Mustang
1970 Chevy Chevelle SS 454
Diamond T620 Semi-Skirted Tanker
1969 Ford Falcon

Nếu không chỉ rõ việc sắp xếp được thực hiện theo thứ tự tăng hay giảm dần, MySQL sẽ mặc định việc sắp xếp dữ liệu được thực hiện theo thứ tự tăng dần.

6. Kết hợp các kết quả với toán tử UNION

UNION cho phép kết hợp hai hoặc nhiều bộ kết quả từ nhiều bảng với nhau. Cú pháp của việc sử dụng MySQL UNION là như sau:

```
SELECT statement
UNION [DISTINCT | ALL]
SELECT statement
UNION [DISTINCT | ALL]
...
```

Để sử dụng UNION, có một số nguyên tắc cần phải làm theo:

- Số lượng các cột trong mỗi câu lệnh SELECT phải giống nhau.
- Các kiểu dữ liệu của cột trong danh sách cột của câu lệnh SELECT phải giống nhau hoặc ít nhất là có thể chuyển đổi sang cho nhau.

Theo mặc định, UNION MySQL loại bỏ tất cả các hàng trùng lặp từ kết quả ngay cả khi không sử dụng từ khoá DISTINCT sau từ khoá UNION.

Nếu sử dụng UNION ALL, các hàng trùng lặp vẫn còn trong tập hợp kết quả cuối cùng. chỉ nên sử dụng điều này trong các trường hợp hoặc là muốn giữ lại bản sao các hàng, hoặc chắc chắn rằng có không có bản sao các hàng trong tập hợp kết quả.
Ví dụ: kết hợp thông tin về các khách hàng và nhân viên thành một tập hợp kết quả, sử dụng truy vấn sau đây:

```
SELECT customerNumber id, contactLastname name
FROM customers
UNION
SELECT employeeNumber id,firstname name
FROM employees
```

id	name
103	Schmitt
112	King
114	Ferguson
119	Labrune
121	Bergulfsen
124	Nelson
125	Piestrzeniewicz
128	Keitel
129	Murphy
131	Lee
141	Freyre
144	Berglund
145	Petersen
146	Saveley
148	Natividad
151	Young

Khi sử dụng ORDER BY để sắp xếp kết quả với UNION, phải đặt nó ở vị trí cuối cùng trong mệnh đề SELECT.

Ví dụ: Giả sử kết hợp thông tin của nhân viên và khách hàng, sau đó muốn sắp xếp kết quả theo ***tên*** và ***ID*** thứ tự tăng dần

```
(SELECT customerNumber, contactLastname
FROM customers)
UNION
(SELECT employeeNumber, firstname
FROM employees)
ORDER BY contactLastname, customerNumber
```

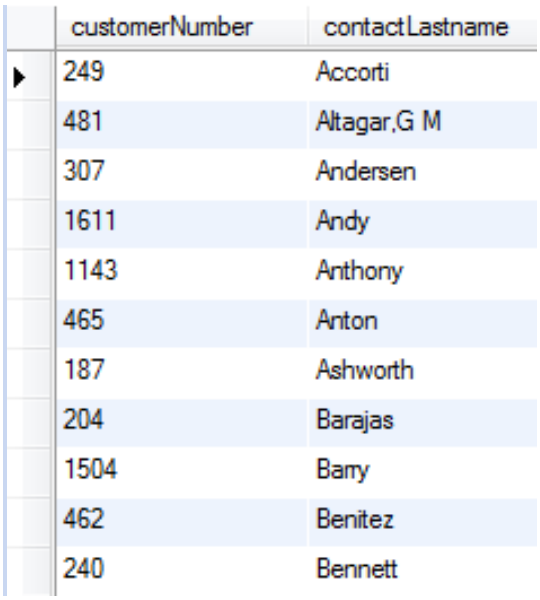
customerNumber	contactLastname
249	Accorti
481	Altagar,G M
307	Andersen
1611	Andy
1143	Anthony
465	Anton
187	Ashworth
204	Barajas
1504	Barry
462	Benitez
240	Bennett
144	Berglund
121	Bergulfsen
172	Bertrand
321	Brown
324	Brown

Nếu tên cột không giống nhau trong hai mệnh đề SELECT của phép UNION, tên nào sẽ được hiển thị ở đầu ra nếu chúng ta không sử dụng bí danh cho mỗi cột trong mệnh đề SELECT. Câu trả lời là MySQL sẽ sử dụng các tên cột của câu lệnh SELECT đầu tiên là tên cột trong kết quả đầu ra

```
(SELECT customerNumber, contactLastname
FROM customers)
UNION
```

```
(SELECT employeeNumber, firstname
FROM employees)
ORDER BY contactLastname, customerNumber
```

Kết quả của phép toán hợp giữa hai tập kết quả từ bảng dữ liệu customers và employees



customerNumber	contactLastname
249	Accorti
481	Altagar,G M
307	Andersen
1611	Andy
1143	Anthony
465	Anton
187	Ashworth
204	Barajas
1504	Barry
462	Benitez
240	Bennett

MySQL cũng cung cấp một lựa chọn khác để sắp xếp các kết quả thiết lập dựa trên vị trí cột trong mệnh đề ORDER BY như truy vấn sau đây:

```
(SELECT customerNumber, contactLastname
FROM customers)
UNION
(SELECT employeeNumber,firstname
FROM employees)
ORDER BY 2, 1
```

customerNumber	contactLastname
249	Accorti
481	Altagar,G M
307	Andersen
1611	Andy
1143	Anthony
465	Anton
187	Ashworth
204	Barajas
1504	Barry
462	Benitez
240	Bennett
144	Berglund
121	Bergulfsen
172	Bertrand

❖ Bài tập thực hành:

1. Dùng toán tử IN để đưa ra thông tin của các khách hàng sống tại các thành phố Nantes và Lyon.
2. Sử dụng BETWEEN để tìm các đơn hàng đã được chuyển trong khoảng thời gian từ ‘10/1/2003’ đến ‘10/3/2003’.
3. Sử dụng LIKE để đưa ra thông tin về các nhóm hàng hoá có chứa từ ‘CARS’.
4. Truy vấn 10 sản phẩm có số lượng trong kho là lớn nhất.
5. Đưa ra danh sách các sản phẩm và thêm thuộc tính là tiền hàng tồn của sản phẩm.

Bài thực hành số 5

Các hàm xử lý của MySQL

❖ ***Nội dung chính*** : Trong bài này, chúng ta sẽ làm quen với một số hàm (functions) cơ bản:

- Hàm xử lý xâu kí tự: Substring, Concat, Replace
- Hàm điều kiện If
- Hàm LAST_INSERT_ID
- Hàm xử lý thời gian: DATEDIFF, ADDDATE, EXTRACT

1. Hàm xử lý chuỗi SUBSTRING

Hàm Substring cho phép trích xuất một chuỗi con từ một chuỗi khác, bắt đầu tại vị trí cụ thể và với một độ dài nhất định. Sau đây minh họa các hình thức sử dụng khác nhau của hàm này.

```
SUBSTRING(str,pos);
SUBSTRING(str FROM pos);
```

Kết quả của câu lệnh ở trên trả về một chuỗi con từ một chuỗi ***str*** bắt đầu từ vị trí ***pos***

```
SUBSTRING(str,pos,len);
SUBSTRING(str FROM pos FOR len);
```

Hai câu lệnh ở trên trả về một chuỗi con từ một chuỗi ***str***, bắt đầu tại vị trí ***pos*** và chuỗi con trả về chỉ có ***len*** ký tự. Lưu ý rằng FROM là từ khoá cú pháp SQL chuẩn. Chúng ta hãy xem xét một số ví dụ sau"

```
SELECT substring('MySQL Substring',7);
```
Trả về: `Substring`
```
SELECT substring('MySQL Substring' FROM 7);
```
Trả về: `Substring`
```
SELECT substring('MySQL Substring',7,3);
```
Trả về: `Sub`
```
SELECT substring('MySQL Substring' FROM 7 FOR 3);
```

Trả về: `Sub`

cũng có thể sử dụng giá trị âm cho tham số pos. Nếu sử dụng giá trị âm cho tham số pos, sự bắt đầu của chuỗi con được tính từ cuối của chuỗi, ví dụ

```
SELECT substring('MySQL Substring',-9);
```

Trả về: `Substring`

Đôi khi thấy đoạn mã sử dụng *substr ()* thay vì hàm *substring ()* . *Substr* là từ đồng nghĩa với *substring,* vì vậy nó có tác dụng tương tự.

2. Hàm CONCAT

Hàm Concat được sử dụng để nối hai hoặc nhiều chuỗi. Nếu các đối số là số, chúng sẽ được chuyển đổi thành chuỗi trước khi nối. Nếu bất kỳ đối số trong danh sách đối số là NULL, hàm concat sẽ trả về NULL.

```
CONCAT(str1,str2,...)
```

Ví dụ: Để hiển thị tên đầy đủ đầu tiên của địa chỉ liên lạc của khách hàng chúng tôi sử dụng hàm concat để nối các tên đầu tiên và tên cuối cùng và dấu phân cách giữa chúng. Dưới đây là truy vấn:

```
SELECT CONCAT(contactLastname,', ',contactFirstname)
fullname
FROM customers
```

fullname
Schmitt, Carine
King, Jean
Ferguson, Peter
Labrune, Janine
Bergulfsen, Jonas
Nelson, Susan
Piestrzeniewicz, Zbyszek
Keitel, Roland
Murphy, Julie
Lee, Kwai
Freyre, Diego
Berglund, Christina
Petersen, Jytte
Saveley, Mary
Natividad, Eric
Young, Jeff

MySQL cũng hỗ trợ hàm concat_ws cho phép chúng ta nối hai hay nhiều hơn hai chuỗi với một dấu phân cách được xác định trước. Cú pháp của hàm concat_ws là:

```
CONCAT_WS(seperator,str1,str2,...)
```

Tham số đầu tiên là dấu phân cách do định nghĩa và sau đó là những chuỗi muốn nối. Kết quả trả về là một chuỗi đã được ghép nối,với dấu phân cách giữa mỗi thành phần ghép nối. Có thể đạt được kết quả tương tự trong ví dụ trên bằng cách sử dụng concat_ws thay vì hàm concat.

```
SELECT CONCAT_WS('; ',contactLastname,contactFirstname)
fullname
FROM customers
```

fullname
Schmitt; Carine
King; Jean
Ferguson; Peter
Labrune; Janine
Bergulfsen; Jonas
Nelson; Susan
Piestrzeniewicz; Zbyszek
Keitel; Roland
Murphy; Julie
Lee; Kwai
Freyre; Diego
Berglund; Christina
Petersen; Jytte
Saveley; Mary
Natividad; Eric
Young; Jeff

Dưới đây là một ví dụ khác của việc sử dụng concat_ws để có được định dạng địa chỉ của khách hàng.

```
SELECT CONCAT_WS(char(10),
           CONCAT_WS(' ',contactLastname,contactFirstname),
addressLine1, addressLine2,
              CONCAT_WS(' ',postalCode,city),  country,
              CONCAT_WS(char(10),'')) AS Customer_Address
FROM customers
```

Customer_Address
Schmitt Carine 54, rue Royale44000 NantesFrance
King Jean8489 Strong St.83030 Las VegasUSA
Ferguson Peter636 St Kilda RoadLevel 33004 MelbourneAustralia
Labrune Janine 67, rue des Cinquante Otages44000 NantesFrance
Bergulfsen Jonas Erling Skakkes gate 784110 StavernNorway
Nelson Susan5677 Strong St.97562 San RafaelUSA
Piestrzeniewicz Zbyszek ul. Filtrowa 6801-012 WarszawaPoland
Keitel RolandLyonerstr. 3460528 FrankfurtGermany
Murphy Julie5557 North Pendale Street94217 San FranciscoUSA
Lee Kwai897 Long Airport Avenue10022 NYCUSA
Freyre Diego C/ Moralzarzal, 8628034 MadridSpain
Berglund Christina Berguvsvägen 8S-958 22 LuleåSweden
Petersen Jytte Vinbæltet 341734 KobenhavnDenmark
Saveley Mary 2, rue du Commerce69004 LyonFrance
Natividad EricBronz Sok.Bronz Apt. 3/6 Tesvikiye079903 SingaporeSingapore
Young Jeff4092 Furth CircleSuite 40010022 NYCUSA

3. Hàm REPLACE

MySQL cung cấp cho một hàm xử lý chuỗi hữu ích là Replace, cho phép thay thế một chuỗi trong một cột của một bảng bằng một chuỗi mới.

Cú pháp của hàm như sau:

```
UPDATE <tên bảng>
SET tên cột = REPLACE(tên cột,xâu cần tìm,xâu thay thế)
WHERE <các điều kiện>
```

Lưu ý: rằng khi tìm kiếm các văn bản để thay thế, MySQL có phân biệt chữ hoa và chữ thường.

Ví dụ: nếu muốn sửa lỗi chính tả trong bảng Product trong cơ sở dữ liệu mẫu, sử dụng hàm Replace như sau:

```
UPDATE products
SET productDescription =
REPLACE(productDescription,'abuot','about')
```

Truy vấn sẽ xem xét cột productDescription và tìm thấy tất cả các lần xuất hiện của lỗi chính tả 'abuot' và thay thế nó bằng từ chính xác 'about'.

Điều rất quan trọng cần lưu ý rằng trong hàm Replace, tham số đầu tiên là tên trường không đặt trong dấu ‘’. Nếu đặt dấu để tên trường như 'field_name’, truy vấn sẽ cập nhật nội dung của cột 'field_name’, gây mất dữ liệu.

Hiện nay hàm Replace không hỗ trợ biểu thức chính quy vì vậy nếu cần phải thay thế một chuỗi văn bản bằng một mẫu, cần phải sử dụng hàm do người dùng định nghĩa (UDF) từ thư viện bên ngoài.

4. Hàm IF

IF là một hàm điều khiển, trả về kết quả là một chuỗi hoặc số dựa trên một điều kiện cho trước. Cú pháp của hàm IF như sau:

```
IF(expr,if_true_expr,if_false_expr)
```

- Tham số đầu tiên là expr sẽ được kiểm tra là đúng hay sai. Giá trị thực có nghĩa là expr không bằng 0 và expr không bằng NULL. Lưu ý rằng NULL là một giá trị đặc biệt, không bằng bất cứ điều gì khác, ngay cả bản thân nó.
- Nếu expr được đánh giá là đúng, hàm IF sẽ trả lại if_true_expr, nếu không nó sẽ trả lại if_false_expr.

Ví dụ:

```
SELECT IF(1 = 2,'true','false');
Trả về: false
SELECT IF(1 = 1,' true','false');
Trả về: true
```

Ví dụ: Trong bảng khách hàng, không phải tất cả các khách hàng đều có thông tin về state. Vì vậy, khi chúng ta lựa chọn khách hàng, thông tin state sẽ hiển thị giá trị NULL, không có ý nghĩa cho mục đích báo cáo.

```
SELECT customerNumber,
       customerName,
```

```
        state,
        country
FROM customers;
```

customerNumber	customerName	state	country
103	Atelier graphique	NULL	France
112	Signal Gift Stores	NV	USA
114	Australian Collectors, Co.	Victoria	Australia
119	La Rochelle Gifts	NULL	France
121	Baane Mini Imports	NULL	Norway
124	Mini Gifts Distributors Ltd.	CA	USA
125	Havel & Zbyszek Co	NULL	Poland
128	Blauer See Auto, Co.	NULL	Germany
129	Mini Wheels Co.	CA	USA
131	Land of Toys Inc.	NY	USA
141	Euro+ Shopping Channel	NULL	Spain
144	Volvo Model Replicas, Co	NULL	Sweden
145	Danish Wholesale Imports	NULL	Denmark
146	Saveley & Henriot, Co.	NULL	France
148	Dragon Souveniers, Ltd.	NULL	Singapore
151	Muscle Machine Inc	NY	USA

Chúng ta có thể sử dụng IF để hiển thị trạng thái của khách hàng là N / A nếu nó là NULL như sau:

```
SELECT customerNumber,
        customerName,
        IF(state IS NULL,'N/A',state) state,
        country
FROM customers;
```

customerNumber	customerName	state	country
103	Atelier graphique	N/A	France
112	Signal Gift Stores	NV	USA
114	Australian Collectors, Co.	Victoria	Australia
119	La Rochelle Gifts	N/A	France
121	Baane Mini Imports	N/A	Norway
124	Mini Gifts Distributors Ltd.	CA	USA
125	Havel & Zbyszek Co	N/A	Poland
128	Blauer See Auto, Co.	N/A	Germany
129	Mini Wheels Co.	CA	USA
131	Land of Toys Inc.	NY	USA
141	Euro+ Shopping Channel	N/A	Spain
144	Volvo Model Replicas, Co	N/A	Sweden
145	Danish Wholesale Imports	N/A	Denmark
146	Saveley & Henriot, Co.	N/A	France
148	Dragon Souveniers, Ltd.	N/A	Singapore

Ví dụ: Hàm IF cũng rất hữu ích với chức năng tổng hợp. Giả sử nếu muốn biết có bao nhiêu đơn đặt hàng đã vận chuyển và huỷ bỏ cùng một lúc, chúng ta có thể sử dụng IF để đếm như sau:

```
SELECT SUM(IF(status = 'Shipped',1,0))    AS Shipped,
     SUM(IF(status = 'Cancelled',1,0)) AS Cancelled
FROM orders;
```

Shipped	Cancelled
303	6

Trong truy vấn trên, nếu tình trạng của đơn đặt hàng là SHIPPED hoặc CANCELLED, IF sẽ trả lại giá trị 1, nếu không nó trả về 0. Và sau đó hàm SUM sẽ tính toán tổng số để vận chuyển và bị hủy bỏ dựa trên giá trị trả về của hàm IF.

5. Hàm LAST_INSERT_ID

Hàm LAST_INSERT_ID trả về ID của bản ghi cuối cùng được chèn vào bảng, với điều kiện đólà ID của cột có thuộc tính AUTO_INCREMENT.Trong thiết kế cơ sở dữ liệu, thường sử dụng một cột tự động tăng AUTO_INCREMENT. Khi chèn một bản ghi mới

vào bảng có cột AUTO_INCREMENT, MySQL tạo ra ID cho tự động dựa trên các thiết lập của cột đó. có thể có được ID này bằng cách sử dụng hàm LAST_INSERT_ID.

Ví dụ: tạo ra một bảng mới để thử nghiệm được gọi là TBL. Trong bảng TBL, chúng ta sử dụng ID là cột AUTO_INCREMENT.

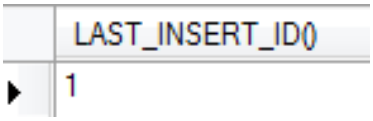
```
CREATE TABLE tbl(
     id INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
        description varchar(250) NOT NULL
);
```

Sau đó, chúng ta sử dụng hàm LAST_INSERT_ID () để có được ID mới chèn.

```
INSERT INTO tbl(description)
VALUES('MySQL last_insert_id');
```

Thực hiện truy vấn:

```
SELECT LAST_INSERT_ID();
```



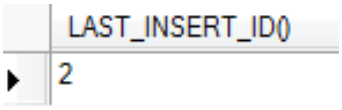
LAST_INSERT_ID()
1

Điều quan trọng cần lưu ý rằng nếu chèn nhiều bản ghi vào bảng bằng cách sử dụng câu lệnh INSERT duy nhất, hàm LAST_INSERT_ID sẽ trả lại giá trị tạo ra cho các bản ghi chèn vào đầu tiên. Hãy thử các bước sau:

```
INSERT INTO tbl(description)
VALUES('record 1'),
     ('record 2'),
     ('record 3');
```

Thực hiện truy vấn:

```
SELECT LAST_INSERT_ID();
```



Chúng ta đã chèn 3 bản ghi bằng cách sử dụng câu lệnh INSERT và hàm LAST_INSERT_ID trả lại ID của bản ghi đầu tiên như mong muốn. MySQL LAST_INSERT_ID hoạt động dựa trên nguyên tắc độc lập với client. Nó có nghĩa là giá trị được trả về bởi hàm LAST_INSERT_ID cho một client cụ thể là giá trị mà client đó tạo ra. Điều này đảm bảo rằng mỗi client có thể nhận được ID riêng của mình mà không cần phải quan tâm đến các hoạt động của các client khác và không cần sử dụng cơ chế lock hay transaction (sẽ học sau).

6. Hàm DATEDIFF

Trong một số trường hợp, cần phải tính toán số ngày giữa hai mốc thời gian, ví dụ số ngày từ ngày vận chuyển và ngày yêu cầu trong một đơn đặt hàng. Trong những trường hợp này, cần phải sử dụng hàm DATEDIFF.
Cú pháp DATEDIFF như sau:

```
DATEDIFF(expr1,expr2)
```

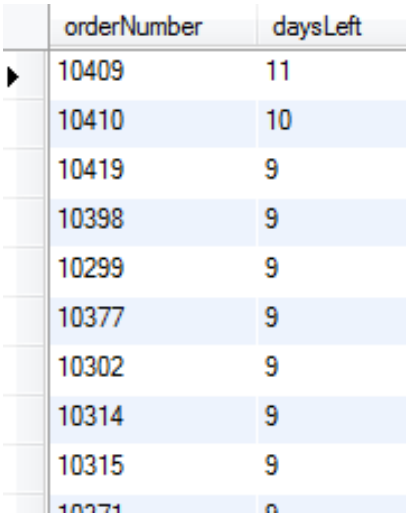
expr1 và *expr2* là hai mốc thời gian.
Ví dụ:

```
SELECT DATEDIFF('2011-08-17','2011-08-17');
Trả về: 0 day

SELECT DATEDIFF('2011-08-17','2011-08-08');
Trả về: 9 days

SELECT DATEDIFF('2011-08-08','2011-08-17');
Trả về: 9 days
```

Ví dụ: Để tính toán số ngày còn lại giữa ngày vận chuyển và ngày yêu cầu để trong đơn đặt hàng, chúng ta sử dụng DATEDIFF như sau:

```
SELECT orderNumber,
DATEDIFF(requiredDate,shippedDate) AS daysLeft
```

```
FROM orders
ORDER BY daysLeft DESC;
```



orderNumber	daysLeft
10409	11
10410	10
10419	9
10398	9
10299	9
10377	9
10302	9
10314	9
10315	9

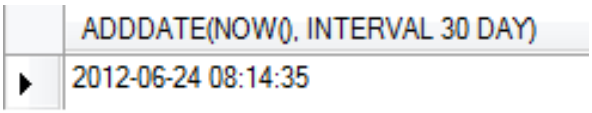
7. Hàm ADDDATE, EXTRACT

MySQL cũng hỗ trợ một số hàm xử lý ngày tháng khác như: ADDDATE, EXTRACT

Hàm ADDDATE: trả về một giá trị thời gian là kết quả của thao tác trên một giá trị thời gian khác.

Ví dụ: đưa ra ngày tháng sau ngày giờ hiện tại 30 ngày:

```
SELECT ADDDATE(NOW(), INTERVAL 30 DAY);
```



ADDDATE(NOW(), INTERVAL 30 DAY)
2012-06-24 08:14:35

Sử dụng từ khóa DAY để chỉ giá trị sẽ cộng vào là ngày.

Ví dụ: đưa ra các đơn đặt hàng trong khoảng 30 ngày tính từ ngày 1/5/2005

```
SELECT *
FROM orders
WHERE orderDate>= '2005-5-1' AND orderDate < ADDDATE('2005-
5-1', INTERVAL 30 DAY);
```

Kết quả truy vấn:

orderNumber	orderDate	requiredDate	shippedDate	status	comments
10411	2005-05-01 00:00:00	2005-05-08 00:00:00	2005-05-06 00:00:00	Shipped	NULL
10412	2005-05-03 00:00:00	2005-05-13 00:00:00	2005-05-05 00:00:00	Shipped	NULL
10413	2005-05-05 00:00:00	2005-05-14 00:00:00	2005-05-09 00:00:00	Shipped	Customer requested that DHL is used for this shipping
10414	2005-05-06 00:00:00	2005-05-13 00:00:00	NULL	On Hold	Customer credit limit exceeded. Will ship when a payment is received.
10415	2005-05-09 00:00:00	2005-05-20 00:00:00	2005-05-12 00:00:00	Disputed	Customer claims the scales of the models don't match what was discuss
10416	2005-05-10 00:00:00	2005-05-16 00:00:00	2005-05-14 00:00:00	Shipped	NULL
10417	2005-05-13 00:00:00	2005-05-19 00:00:00	2005-05-19 00:00:00	Disputed	Customer doesn't like the colors and precision of the models.
10418	2005-05-16 00:00:00	2005-05-24 00:00:00	2005-05-20 00:00:00	Shipped	NULL
10419	2005-05-17 00:00:00	2005-05-28 00:00:00	2005-05-19 00:00:00	Shipped	NULL
10420	2005-05-29 00:00:00	2005-06-07 00:00:00	NULL	In Process	NULL
10421	2005-05-29 00:00:00	2005-06-06 00:00:00	NULL	In Process	Custom shipping instructions were sent to warehouse

Ví dụ: đưa ra các đơn đặt hàng tính từ trước ngày 1/5/2005, 30 ngày đến ngày 1/5/2005

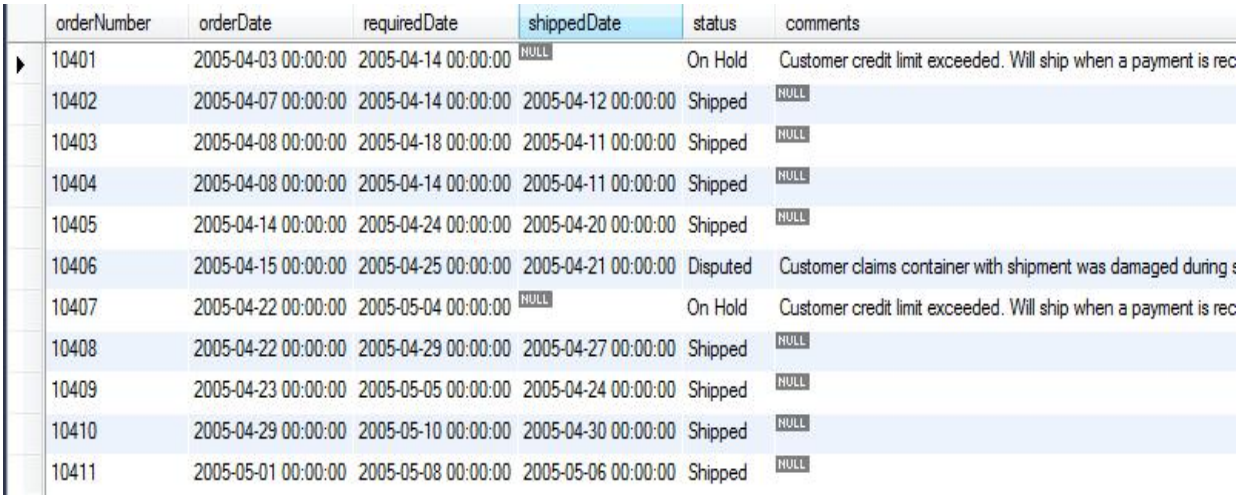
```
SELECT *
FROM orders
WHERE orderDate<= '2005-5-1' AND orderDate > ADDDATE('2005-
5-1', INTERVAL -30 DAY);
```

orderNumber	orderDate	requiredDate	shippedDate	status	comments
10401	2005-04-03 00:00:00	2005-04-14 00:00:00	NULL	On Hold	Customer credit limit exceeded. Will ship when a payment is rece
10402	2005-04-07 00:00:00	2005-04-14 00:00:00	2005-04-12 00:00:00	Shipped	NULL
10403	2005-04-08 00:00:00	2005-04-18 00:00:00	2005-04-11 00:00:00	Shipped	NULL
10404	2005-04-08 00:00:00	2005-04-14 00:00:00	2005-04-11 00:00:00	Shipped	NULL
10405	2005-04-14 00:00:00	2005-04-24 00:00:00	2005-04-20 00:00:00	Shipped	NULL
10406	2005-04-15 00:00:00	2005-04-25 00:00:00	2005-04-21 00:00:00	Disputed	Customer claims container with shipment was damaged during sh
10407	2005-04-22 00:00:00	2005-05-04 00:00:00	NULL	On Hold	Customer credit limit exceeded. Will ship when a payment is rece
10408	2005-04-22 00:00:00	2005-04-29 00:00:00	2005-04-27 00:00:00	Shipped	NULL
10409	2005-04-23 00:00:00	2005-05-05 00:00:00	2005-04-24 00:00:00	Shipped	NULL
10410	2005-04-29 00:00:00	2005-05-10 00:00:00	2005-04-30 00:00:00	Shipped	NULL
10411	2005-05-01 00:00:00	2005-05-08 00:00:00	2005-05-06 00:00:00	Shipped	NULL

Nếu thời gian cộng vào là tháng, năm thì từ khóa tương ứng được sử dụng là MONTH, YEAR.

Ví dụ trên có thể viết lại như sau

```
SELECT *
FROM orders
WHERE orderDate<= '2005-5-1' AND orderDate > ADDDATE('2005-
5-1', INTERVAL -1 MONTH);
```

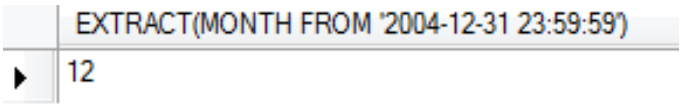


orderNumber	orderDate	requiredDate	shippedDate	status	comments
10401	2005-04-03 00:00:00	2005-04-14 00:00:00	NULL	On Hold	Customer credit limit exceeded. Will ship when a payment is rec
10402	2005-04-07 00:00:00	2005-04-14 00:00:00	2005-04-12 00:00:00	Shipped	NULL
10403	2005-04-08 00:00:00	2005-04-18 00:00:00	2005-04-11 00:00:00	Shipped	NULL
10404	2005-04-08 00:00:00	2005-04-14 00:00:00	2005-04-11 00:00:00	Shipped	NULL
10405	2005-04-14 00:00:00	2005-04-24 00:00:00	2005-04-20 00:00:00	Shipped	NULL
10406	2005-04-15 00:00:00	2005-04-25 00:00:00	2005-04-21 00:00:00	Disputed	Customer claims container with shipment was damaged during s
10407	2005-04-22 00:00:00	2005-05-04 00:00:00	NULL	On Hold	Customer credit limit exceeded. Will ship when a payment is rec
10408	2005-04-22 00:00:00	2005-04-29 00:00:00	2005-04-27 00:00:00	Shipped	NULL
10409	2005-04-23 00:00:00	2005-05-05 00:00:00	2005-04-24 00:00:00	Shipped	NULL
10410	2005-04-29 00:00:00	2005-05-10 00:00:00	2005-04-30 00:00:00	Shipped	NULL
10411	2005-05-01 00:00:00	2005-05-08 00:00:00	2005-05-06 00:00:00	Shipped	NULL

Hàm EXTRACT: tách ra các giá trị như ngày, tháng, năm từ một giá trị có kiểu thời gian.

Ví dụ: đưa ra tháng của một giá trị thời gian:

```
SELECT EXTRACT(MONTH FROM '2004-12-31 23:59:59');
```

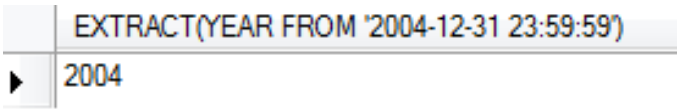


EXTRACT(MONTH FROM '2004-12-31 23:59:59')
12

* *Có thể sử dụng hàm MONTH('2004-12-31 23:59:59') cho ví dụ trên*

Ví dụ: đưa ra tháng của một giá trị thời gian:

```
SELECT EXTRACT(YEAR FROM '2004-12-31 23:59:59');
```

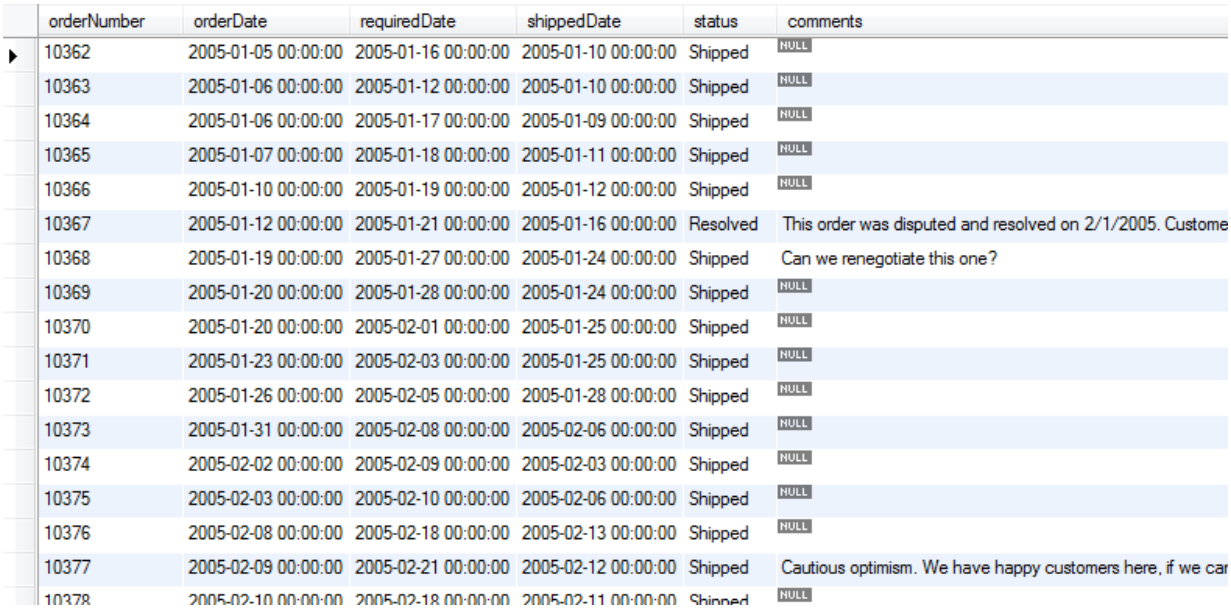


EXTRACT(YEAR FROM '2004-12-31 23:59:59')
2004

Ví dụ: đưa ra các đơn hàng đặt năm 2005

```
SELECT *
```

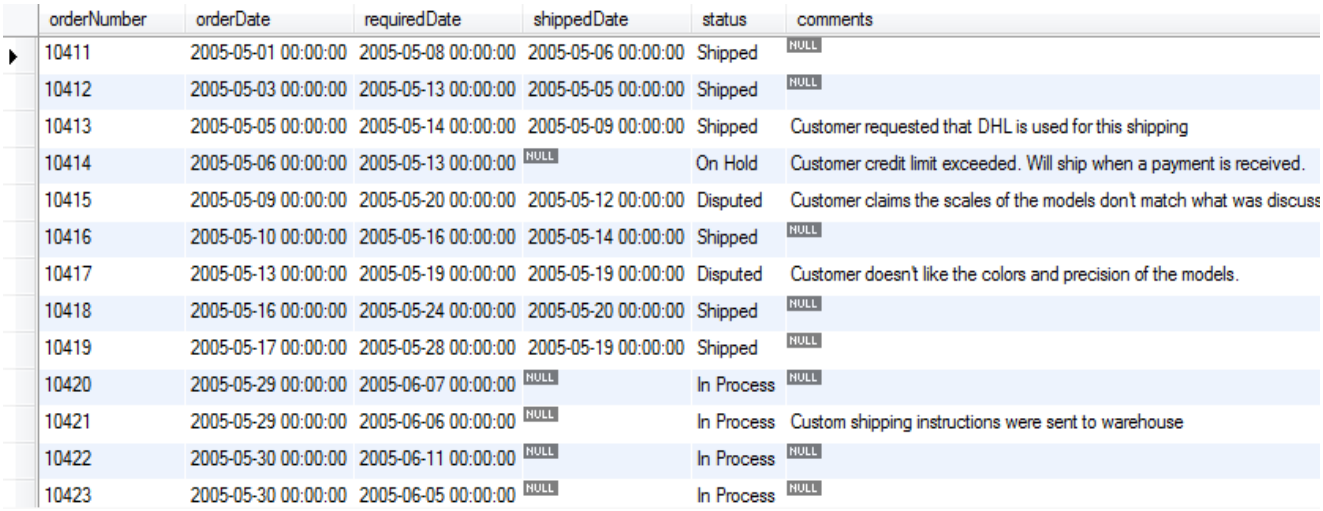
```
FROM orders

WHERE EXTRACT(YEAR FROM orderDate) = 2005
```



orderNumber	orderDate	requiredDate	shippedDate	status	comments
10362	2005-01-05 00:00:00	2005-01-16 00:00:00	2005-01-10 00:00:00	Shipped	NULL
10363	2005-01-06 00:00:00	2005-01-12 00:00:00	2005-01-10 00:00:00	Shipped	NULL
10364	2005-01-06 00:00:00	2005-01-17 00:00:00	2005-01-09 00:00:00	Shipped	NULL
10365	2005-01-07 00:00:00	2005-01-18 00:00:00	2005-01-11 00:00:00	Shipped	NULL
10366	2005-01-10 00:00:00	2005-01-19 00:00:00	2005-01-12 00:00:00	Shipped	NULL
10367	2005-01-12 00:00:00	2005-01-21 00:00:00	2005-01-16 00:00:00	Resolved	This order was disputed and resolved on 2/1/2005. Custome
10368	2005-01-19 00:00:00	2005-01-27 00:00:00	2005-01-24 00:00:00	Shipped	Can we renegotiate this one?
10369	2005-01-20 00:00:00	2005-01-28 00:00:00	2005-01-24 00:00:00	Shipped	NULL
10370	2005-01-20 00:00:00	2005-02-01 00:00:00	2005-01-25 00:00:00	Shipped	NULL
10371	2005-01-23 00:00:00	2005-02-03 00:00:00	2005-01-25 00:00:00	Shipped	NULL
10372	2005-01-26 00:00:00	2005-02-05 00:00:00	2005-01-28 00:00:00	Shipped	NULL
10373	2005-01-31 00:00:00	2005-02-08 00:00:00	2005-02-06 00:00:00	Shipped	NULL
10374	2005-02-02 00:00:00	2005-02-09 00:00:00	2005-02-03 00:00:00	Shipped	NULL
10375	2005-02-03 00:00:00	2005-02-10 00:00:00	2005-02-06 00:00:00	Shipped	NULL
10376	2005-02-08 00:00:00	2005-02-18 00:00:00	2005-02-13 00:00:00	Shipped	NULL
10377	2005-02-09 00:00:00	2005-02-21 00:00:00	2005-02-12 00:00:00	Shipped	Cautious optimism. We have happy customers here, if we car
10378	2005-02-10 00:00:00	2005-02-18 00:00:00	2005-02-11 00:00:00	Shipped	NULL

Ví dụ: đưa ra các đơn hàng đặt trong tháng 5 năm 2005

```
SELECT *
FROM orders
WHERE EXTRACT(YEAR FROM orderDate) = 2005 and EXTRACT(MONTH
FROM orderDate) = 5;
```



orderNumber	orderDate	requiredDate	shippedDate	status	comments
10411	2005-05-01 00:00:00	2005-05-08 00:00:00	2005-05-06 00:00:00	Shipped	NULL
10412	2005-05-03 00:00:00	2005-05-13 00:00:00	2005-05-05 00:00:00	Shipped	NULL
10413	2005-05-05 00:00:00	2005-05-14 00:00:00	2005-05-09 00:00:00	Shipped	Customer requested that DHL is used for this shipping
10414	2005-05-06 00:00:00	2005-05-13 00:00:00	NULL	On Hold	Customer credit limit exceeded. Will ship when a payment is received.
10415	2005-05-09 00:00:00	2005-05-20 00:00:00	2005-05-12 00:00:00	Disputed	Customer claims the scales of the models don't match what was discuss
10416	2005-05-10 00:00:00	2005-05-16 00:00:00	2005-05-14 00:00:00	Shipped	NULL
10417	2005-05-13 00:00:00	2005-05-19 00:00:00	2005-05-19 00:00:00	Disputed	Customer doesn't like the colors and precision of the models.
10418	2005-05-16 00:00:00	2005-05-24 00:00:00	2005-05-20 00:00:00	Shipped	NULL
10419	2005-05-17 00:00:00	2005-05-28 00:00:00	2005-05-19 00:00:00	Shipped	NULL
10420	2005-05-29 00:00:00	2005-06-07 00:00:00	NULL	In Process	NULL
10421	2005-05-29 00:00:00	2005-06-06 00:00:00	NULL	In Process	Custom shipping instructions were sent to warehouse
10422	2005-05-30 00:00:00	2005-06-11 00:00:00	NULL	In Process	NULL
10423	2005-05-30 00:00:00	2005-06-05 00:00:00	NULL	In Process	NULL

❖ Bài tập thực hành:

1. Lấy ra 50 ký tự đầu tiên của phần mô tả sản phẩm, đặt tên là ‘Title of products’
2. Đưa ra mô tả về các nhân viên theo định dạng ‘Fullname, jobTitle.’
3. Thay thế toàn bộ tên nhóm hàng ‘Cars’ thành ‘Automobiles’.
4. Tìm 5 đơn hàng được vận chuyển sớm nhất so với ngày yêu cầu.
5. Đưa ra các đơn đặt hàng trong tháng 5 năm 2005 và có ngày chuyển hàng đến chưa xác định.