

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	1
Cài đặt hệ quản trị CSDL và quản lý CSDL	3
1. Cài đặt hệ quản trị CSDL MySQL Server	3
2. Cấu trúc MySQL Server	7
3. Kết nối tới MySQL server	9
4. Tạo, xóa cơ sở dữ liệu (CSDL)	12
Bài thực hành số 2	14
Các kiểu dữ liệu. Tạo và sửa đổi cấu trúc bảng	14
1. Các kiểu dữ liệu	14
2. Tạo bảng Cơ sở dữ liệu	16
3. Thay đổi cấu trúc bảng	22
4. Xóa bảng	24
❖ Bài tập thực hành	24
Bài thực hành số 3	26
Truy vấn cơ bản (phần 1)	26
1. Cài đặt cơ sở dữ liệu mẫu	26
2. Thực hiện truy vấn với câu lệnh SELECT	27
3. Mệnh đề WHERE	30
4. Kết nối các điều kiện với toán tử AND và OR	31
5. IS NULL: tìm các giá trị không xác định	32
6. Từ khoá DISTINCT	33
7. Giới hạn số lượng kết quả với LIMIT	34
❖ Bài tập thực hành:	36
Bài thực hành số 4	37
Truy vấn cơ bản (phần 2)	37
1. Toán tử IN	37
2. Toán tử BETWEEN	38
3. Toán tử LIKE	40
4. Thuộc tính suy diễn (Derived Attribute)	44
5. Sắp xếp kết quả với ORDER BY	45
6. Kết hợp các kết quả với toán tử UNION	47

❖ Bài tập thực hành:	51
Bài thực hành số 5	52
Các hàm xử lý của MySQL	52
1. Hàm xử lý chuỗi SUBSTRING	52
2. Hàm CONCAT	53
3. Hàm REPLACE	56
4. Hàm IF	57
5. Hàm LAST_INSERT_ID	59
6. Hàm DATEDIFF	61
7. Hàm ADDDATE, EXTRACT	62
❖ Bài tập thực hành:	66
Bài thực hành số 6	67
Truy vấn nhóm	67
1. Các hàm nhóm	67
2. Mệnh đề nhóm GROUP BY	69
3. Mệnh đề điều kiện HAVING	73
❖ Bài tập thực hành	75
Bài thực hành số 7	76
Các phép nối bảng dữ liệu	76
1. PHÉP NỐI TRONG (INNER JOIN)	76
2. PHÉP NỐI TRÁI (LEFT JOIN)	83
3. PHÉP TỰ NỐI (Self Join)	87
❖ Bài tập thực hành:	88
Bài thực hành số 8	89
Truy vấn con (Subquery)	89
1. Khái niệm truy vấn con	89
2. Truy vấn con không tương quan	89
3. Truy vấn con tương quan	91
4. Sử dụng truy vấn con	92
❖ Bài tập thực hành	95
Bài thực hành số 9	96
Thêm, sửa, xóa dữ liệu trong bảng	96

1. Câu lệnh INSERT.....	96
2. Câu lệnh UPDATE.....	99
3. Câu lệnh DELETE	100
4. Cập nhật dữ liệu có ràng buộc.....	102
❖ Bài tập thực hành	104
Bài thực hành số 10.....	105
Mô hình hóa CSDL sử dụng công cụ MySQL Workbench.....	105
1. Giới thiệu MySQL Workbench.....	105
2. Tạo mô hình quan hệ thực thể EER	106
3. Tạo CSDL từ mô hình quan hệ thực thể EER.....	113
4. Đồng bộ hóa mô hình EER với CSDL trong MySQL Server.....	114
5. Tạo mô hình quan hệ thực thể EER từ CSDL có sẵn	116
❖ Bài tập thực hành	119

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, phân tích 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 công 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 mình họa quá trình cài đặt trên hệ điều hành MS Windows.



Cài đặt trên hệ điều hành MS Windows

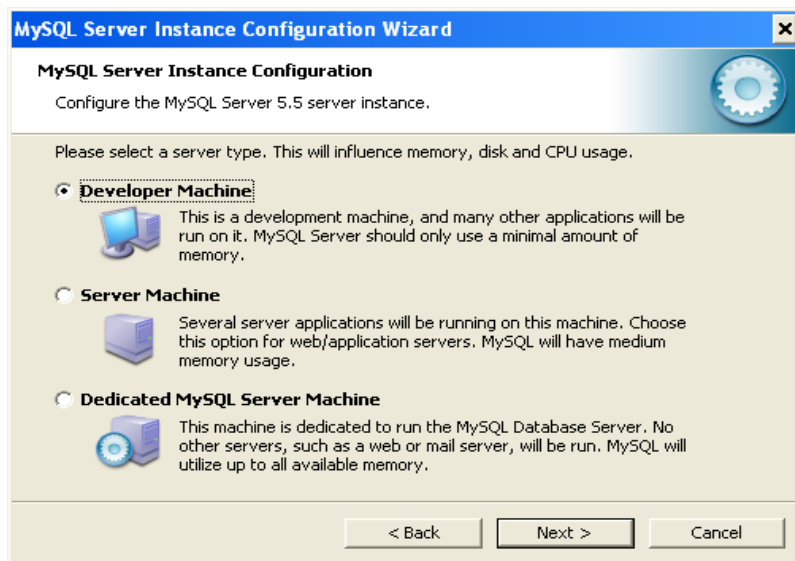
Sau khi thực hiện trình cài đặt trên Window, quá trình cài đặt MySQL Server bắt đầu qua các bước sau:

Bước 1: Lựa chọn kiểu server

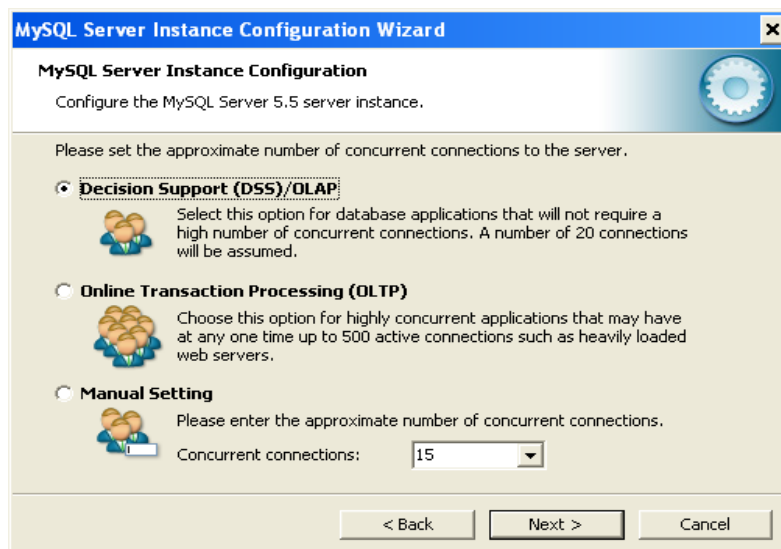
Chúng ta có thể lựa chọn 1 trong 3 kiểu server sau:

- 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.



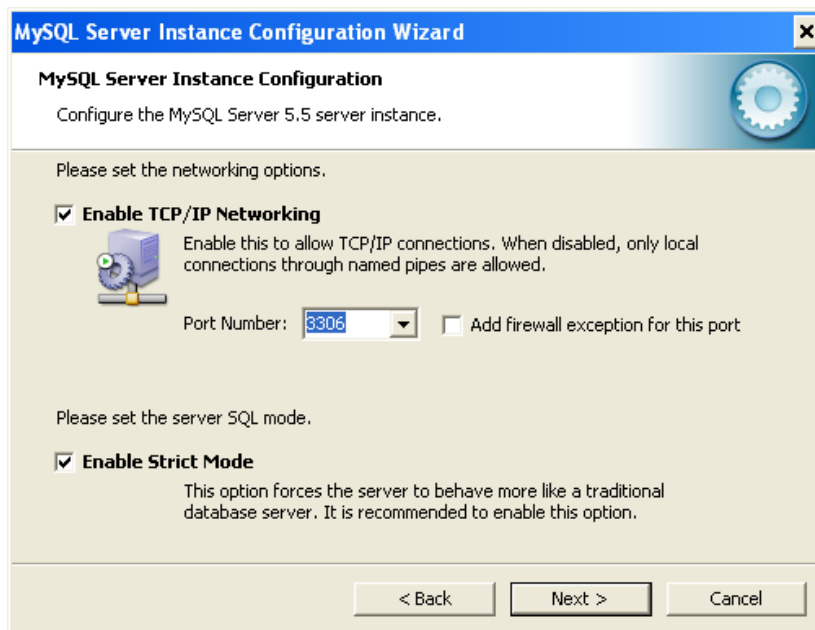
Bước 2: Cấu hình số lượng kết nối đồng thời



- 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 ký 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 MySQL 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 diện dòng lệnh command line](#).

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

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

```

C:\Windows\system32\cmd.exe - mysqld.exe --console
InnoDB: a new database to be created!
120901 22:16:44 InnoDB: Setting file .\ibdata1 size to 10 MB
InnoDB: Database physically writes the file full: wait...
120901 22:16:45 InnoDB: Log file .\ib_logfile0 did not exist: new to be created
InnoDB: Setting log file .\ib_logfile0 size to 5 MB
InnoDB: Database physically writes the file full: wait...
120901 22:16:45 InnoDB: Log file .\ib_logfile1 did not exist: new to be created
InnoDB: Setting log file .\ib_logfile1 size to 5 MB
InnoDB: Database physically writes the file full: wait...
InnoDB: Doublewrite buffer not found: creating new
InnoDB: Doublewrite buffer created
InnoDB: 127 rollback segment(s) active.
InnoDB: Creating foreign key constraint system tables
InnoDB: Foreign key constraint system tables created
120901 22:16:46 InnoDB: Waiting for the background threads to start
120901 22:16:47 InnoDB: 1.1.8 started; log sequence number 0
120901 22:16:47 [Note] Server hostname (bind-address): '0.0.0.0'; port: 3306
120901 22:16:47 [Note] - '0.0.0.0' resolves to '0.0.0.0';
120901 22:16:47 [Note] Server socket created on IP: '0.0.0.0'.
120901 22:16:48 [Note] Event Scheduler: Loaded 0 events
120901 22:16:48 [Note] mysqld.exe: ready for connections.
Version: '5.5.27' socket: '' port: 3306 MySQL Community Server (GPL)

```

Mình 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 <i>localhost</i>)
-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

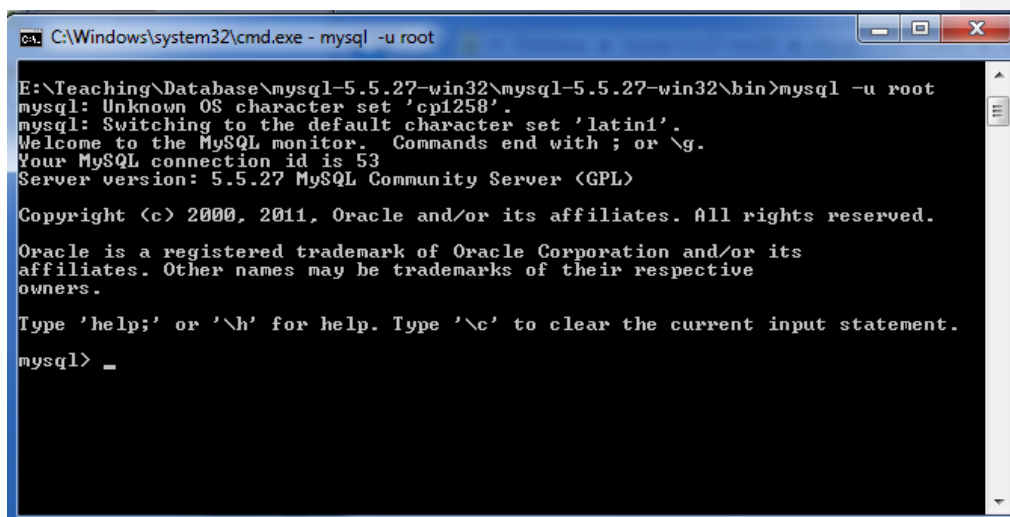
```
shell> basedir\mysql.exe -u user_name -p your_password
```

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..

Ngắt kết nối tới MySQL server sử dụng:

```
mysql> exit;
```

Formatted: Normal

Formatted: Font: (Default) Times New Roman

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 ngỏ định của MySQL server thành 3307 và kết nối tới MySQL server tại cổng này.
2. Thay đổi đường dẫn ngỏ định thư mục chứa CSDL trong file cấu hình
3. Tạo CSDL tên là *my_database*, sau đó dùng lệnh hiển 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 số 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 cố 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
LOB	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 đặc 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 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

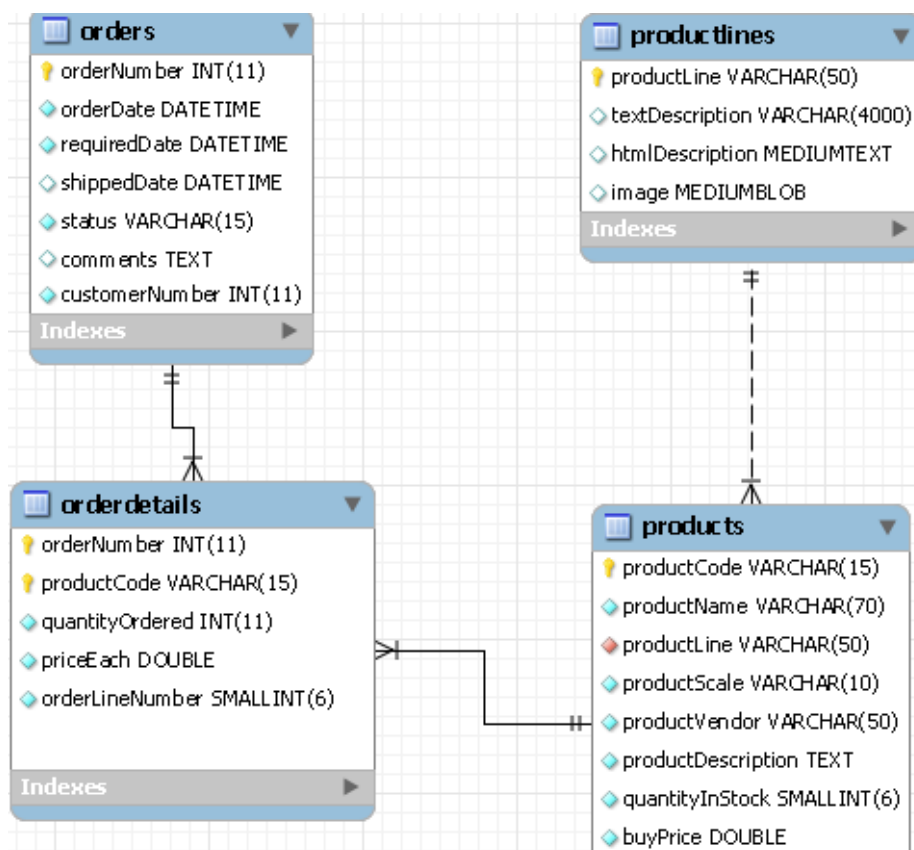
Để xóa bảng khỏi CSDL, sử dụng câu lệnh DROP TABLE:

```
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



Bài thực hành số 3

Truy vấn cơ bản (phần 1)

❖ Nội dung chính

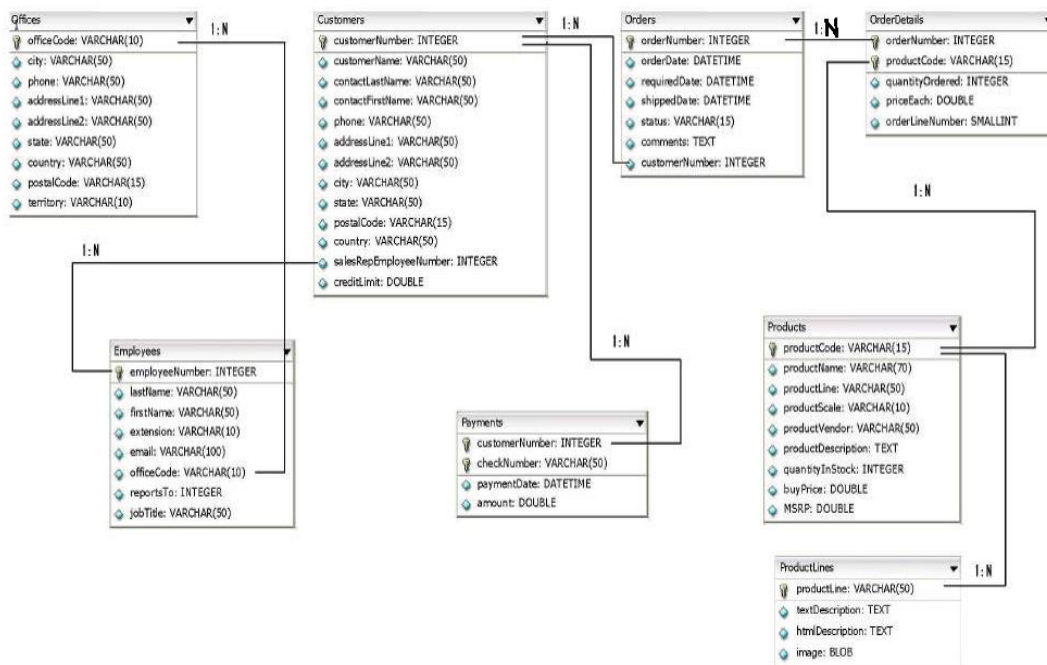
- Câu lệnh Select: cú pháp và cách sử dụng
- Mệnh đề where
- Loại bỏ dữ liệu kết quả trùng lặp với DISTINCT
- Giới hạn các bản ghi trả về bằng LIMIT

1. Cài đặt cơ 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 CSDL về 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 khóa 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 khóa 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	HOLL	President
	1056	Patterson	Mary	x4611	mpatterso@classicmodelcars.com	1	1002	VP Sales
	1076	Firelli	Jeff	x9273	jfirelli@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	Firelli	Julie	x2173	jfirelli@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	ghemande@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 *phép chiếu*.

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 outstanding 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 Souvenirs, Inc	NULL
	Stuttgart Collectable Exch...	NULL

6. Từ khoá DISTINCT

Với từ khoá 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 khoá 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 Caravay	Citeaux	Frédérique	88.60.1555	24, place Kléber	NULL
	242	Alpha Cognac	Roulet	Annette	61.77.6555	1 rue Alsace-Lorraine	NULL
	250	Lyon Souvenirs	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_range 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	Bany

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:

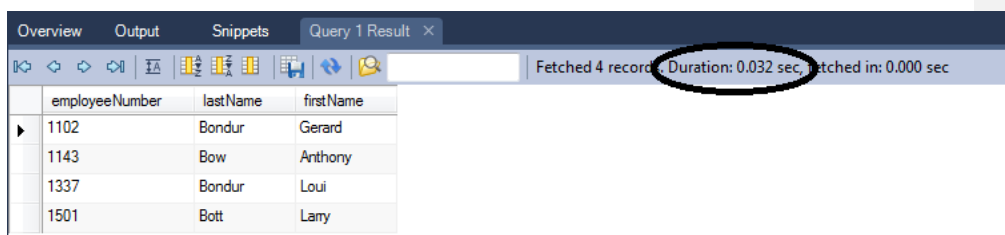
	productCode	productName
▶	S10_2016	1996 Moto Guzzi 1100i
	S24_2000	1960 BSA Gold Star DBD34
	S24_2011	18th century schooner
	S24_2022	1938 Cadillac V-16 Presidential Limousine
	S700_2047	HMS Bounty

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:



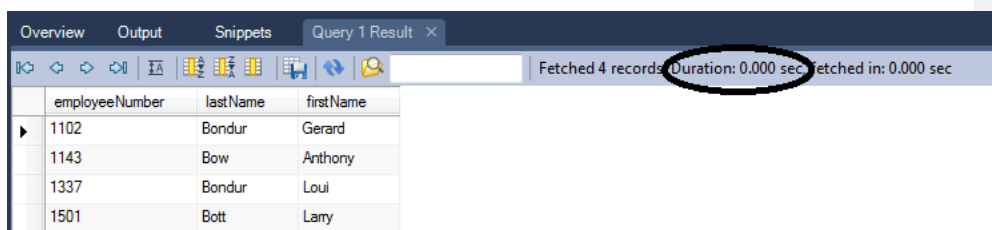
The screenshot shows a MySQL query result window with the following details:

- Tab: Query 1 Result
- Status bar: Fetched 4 records, Duration: 0.032 sec, Fetched in: 0.000 sec
- Table structure:

employeeNumber	lastName	firstName
1102	Bondur	Gerard
1143	Bow	Anthony
1337	Bondur	Loui
1501	Bott	Larry

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%'
```



employeeNumber	lastName	firstName
1102	Bondur	Gerard
1143	Bow	Anthony
1337	Bondur	Loui
1501	Bott	Lamy

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	2400.12

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 Homet 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	Bany
	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	Bary
	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	Bamy
	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
	Piesterzeniewicz, 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
	Labruno Janine 67, rue des Cinquante Otages44000 NantesFrance
	Bergulfsen Jonas Erling Skakkas gate 784110 StavemNorway
	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/ Moralarzal, 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 'abuoat' 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 Souvenirs, 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 Souvenirs, 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 hủy 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 đó là 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();
```


	LAST_INSERT_ID()
▶	2

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
	10371	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 *
```

Formatted: Font: Not Bold

Formatted: Font: Italic

Formatted: Font: Not Bold, Italic

Formatted: Font: Italic

Formatted: Font: Not Bold, Italic

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. Customer
	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 can
	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~~hẹn~~.
5. Đưa ra các đơn đặt hàng trong tháng 5 năm 2005 và có ngày chuyển hàng đến chưa xác định.

Bài thực hành số 6

Truy vấn nhóm

❖ **Nội dung chính:** Trong bài này, chúng ta sẽ làm quen với các hàm nhóm và truy vấn nhóm:

- Các hàm nhóm: SUM, AVG, MAX và MIN , COUNT
- Mệnh đề GROUP BY
- Mệnh đề HAVING

1. Các hàm nhóm

Hàm SUM

Đôi khi các thông tin chúng ta cần không được lưu trữ thực sự trong các bảng cơ sở dữ liệu, nhưng chúng ta có thể lấy được chúng bằng cách tính toán từ dữ liệu được lưu trữ. Ví dụ, chúng ta có bảng *OrderDetails* để lưu trữ thông tin về các đơn đặt hàng. Khi chúng ta nhìn vào đó, chúng ta không biết tổng số tiền của tất cả các sản phẩm bán được là bao nhiêu. Tuy nhiên, hàm tính tổng SUM có thể giúp chúng ta trả lời câu hỏi này. Trước hết chúng ta xem hoạt động của hàm SUM, việc thực hiện nhóm dữ liệu sẽ trình bày trong phần 2

Ví dụ: Tính tổng số lượng hàng hóa hiện còn trong kho

```
SELECT sum(quantityInStock)
FROM products
```

Kết quả trả về như sau:

	sum(quantityInStock)
▶	555131

Hoặc để tính tổng số tiền chúng ta đã thu được từ đầu tới giờ, viết truy vấn như sau:

```
SELECT sum(priceEach * quantityOrdered) total
FROM orderdetails
```

Kết quả trả về như sau:

	total
▶	9857225.609999985

Hàm AVG

AVG được sử dụng để tính giá trị trung bình của một biểu thức, Nó không chấp nhận giá trị NULL. Chúng ta có thể sử dụng AVG để tính toán giá trung bình của tất cả các sản phẩm đã mua như sau:

```
SELECT AVG(buyPrice) average_buy_price
FROM Products
```

Kết quả trả về như sau:

	average_buy_price
▶	54.395181818181825

Hàm MAX và MIN

Hàm MAX trả về giá trị lớn nhất và hàm MIN trả về giá trị nhỏ nhất của một tập các giá trị.

```
MAX(expression)
MIN(expression)
```

Ví dụ: Sử dụng MAX và MIN để lấy ra mức giá cao nhất và mức giá nhỏ nhất của sản phẩm.

```
SELECT MAX(buyPrice) highest_price,
       MIN(buyPrice) lowest_price
FROM Products
```

Kết quả trả về như sau:

	highest_price	lowest_price
▶	103.42	15.91

Hàm COUNT

Hàm COUNT là hàm đếm số lượng, chẳng hạn chúng ta có thể đếm số lượng sản phẩm đang được bán như sau:

```
SELECT COUNT(*) AS Total
FROM products
```

Kết quả trả về như sau:

	Total
▶	110

Lưu ý: một phiên bản khác của hàm COUNT sử dụng tham số là tên cột. Nếu cách này được sử dụng, sẽ chỉ đếm các dòng mà giá trị tại cột đó là khác NULL.

2. Mệnh đề nhóm GROUP BY

Mệnh đề **GROUP BY** được sử dụng để gộp các bản ghi có cùng giá trị tại một hay nhiều cột, thành một tập hợp. GROUP BY nếu có thì nó phải đứng sau mệnh đề WHERE hoặc FROM. Theo sau từ khoá GROUP BY là một danh sách các biểu thức, phân cách nhau bởi dấu phẩy.

```
SELECT col1_, col_2, ... col_n, các hàm nhóm(biểu thức)
FROM tên bảng
WHERE điều kiện
GROUP BY col_1, col_2, ... col_n
ORDER BY danh sách cột
```

Theo định nghĩa, hàm nhóm cho phép chúng ta thực hiện một phép tính trên một tập bản ghi và trả về một giá trị. Hàm nhóm bỏ qua các giá trị null khi thực hiện tính toán, ngoại trừ hàm COUNT. Hàm nhóm thường được sử dụng với mệnh đề GROUP BY của câu lệnh SELECT.

Ví dụ: Giả sử muốn phân chia các đơn đặt hàng theo các nhóm phụ thuộc vào tình trạng của các đơn hàng, có thể làm như sau:

```
SELECT status
FROM orders
GROUP BY status
```

Kết quả trả về như sau:

	status
►	Cancelled
	Disputed
	In Process
	On Hold
	Resolved
	Shipped

Các hàm nhóm được sử dụng với GROUP BY để thực hiện tính toán trên mỗi nhóm các bản ghi và trả về một giá trị duy nhất cho mỗi hàng.

Ví dụ: muốn biết có bao nhiêu đơn đặt hàng trong từng nhóm trạng thái, có thể sử dụng hàm COUNT như sau:

```
SELECT status, count(*)
FROM orders
GROUP BY status
```

Kết quả trả về như sau:

	status	count(*)
►	Cancelled	6
	Disputed	3
	In Process	6
	On Hold	4
	Resolved	4
	Shipped	303

Ví dụ: muốn biết có bao nhiêu loại sản phẩm trong mỗi loại dòng sản phẩm

```
SELECT productLine, count(*)  
FROM products  
GROUP BY productline
```

	productLine	count(*)
▶	Classic Cars	38
	Motorcycles	13
	Planes	12
	Ships	9
	Trains	3
	Trucks and Buses	11
	Vintage Cars	24

Ví dụ: Để có được tổng số tiền cho mỗi sản phẩm đã bán, chúng ta chỉ cần sử dụng chức năng SUM và nhóm sản phẩm. Dưới đây là truy vấn:

```
SELECT productCode, sum(priceEach * quantityOrdered) total  
FROM orderdetails  
GROUP by productCode
```

Kết quả trả về như sau:

	productCode	total
▶	S10_1678	90157.77000000002
	S10_1949	190017.95999999996
	S10_2016	109998.81999999998
	S10_4698	170685.99999999997
	S10_4757	127924.31999999999
	S10_4962	123123.00999999998
	S12_1099	161531.47999999992
	S12_1108	190755.86
	S12_1666	119085.24999999999
	S12_2823	135767.03000000003
	S12_3148	132363.78999999998
	S12_3380	98718.76000000001

Ví dụ: Giả sử chúng ta muốn xem các kết quả của truy vấn trên, hiển thị theo thứ tự tăng dần chúng ta làm như sau:

```
SELECT productCode, sum(priceEach * quantityOrdered) total
FROM orderdetails
GROUP BY productCode
ORDER BY total DESC
```

Kết quả trả về như sau:

	productCode	total
►	S18_3232	276839.98
	S12_1108	190755.86
	S10_1949	190017.95999999996
	S10_4698	170685.99999999997
	S12_1099	161531.47999999992
	S12_3891	152543.02
	S18_1662	144959.90999999997
	S18_2238	142530.62999999998
	S18_1749	140535.60000000003
	S12_2823	135767.03000000003
	S24_3856	134240.71
	S12_3148	132363.78999999998
	S18_2795	132275.97999999998
	S18_4721	130749.31000000001
	S10_4757	127924.31999999999
	S10_4962	123123.00999999998
	S18_4027	122254.75
	S18_3482	121890.6

Lưu ý: sự khác nhau giữa GROUP BY trong MySQL và ANSI SQL

MySQL tuân theo chuẩn ANSI SQL. Tuy nhiên, có 2 sự khác biệt khi sử dụng GROUP BY trong MySQL như sau:

- Trong ANSI SQL, phải thực hiện GROUP BY tất cả các cột xuất hiện trong mệnh đề SELECT. MySQL không đòi hỏi như vậy, có thể đưa thêm các cột vào trong mệnh đề SELECT và không bắt buộc chúng phải xuất hiện ở mệnh đề GROUP BY.

- MySQL cũng cho phép sắp xếp các nhóm theo thứ tự các kết quả tính toán, mặc định là giảm dần.

3. Mệnh đề điều kiện HAVING

HAVING cũng là một mệnh đề có thể xuất hiện hoặc không trong mệnh đề SELECT. Nó chỉ ra một điều kiện lọc trên dữ liệu là một nhóm các bản ghi hoặc là kết quả của việc thực hiện hàm nhóm. HAVING thường được sử dụng cùng với GROUP BY, khi đó điều kiện lọc chỉ được áp dụng trên các cột xuất hiện trong mệnh đề GROUP BY mà thôi. Nếu HAVING không đi kèm với GROUP BY, khi đó nó có ý nghĩa như WHERE mà thôi. Lưu ý rằng, HAVING áp dụng trên các nhóm bản ghi, còn WHERE áp dụng trên từng bản ghi riêng lẻ.

Ví dụ: Chúng ta sử dụng mệnh đề GROUP BY để có được tất cả các đơn đặt hàng, số lượng các mặt hàng bán ra và tổng giá trị trong mỗi đơn đặt hàng như sau:

```
SELECT ordernumber,
       sum(quantityOrdered) AS itemCount,
       sum(priceEach * quantityOrdered) AS total
FROM orderdetails
GROUP BY ordernumber
```

	ordernumber	itemCount	total
▶	10100	151	10223.829999999998
	10101	142	10549.01
	10102	80	5494.78
	10103	541	50218.950000000004
	10104	443	40206.2
	10105	545	53959.21
	10106	675	52151.810000000005
	10107	229	22292.620000000003
	10108	561	51001.219999999994
	10109	212	25833.14
	10110	570	48425.69
	10111	217	16537.850000000002
	10112	52	7674.940000000005
	10113	143	11044.300000000001
	10114	351	33383.140000000001
	10115	210	21665.980000000003
	10116	27	1627.56
	10117	402	44380.15

Bây giờ, có thể yêu cầu hiển thị chỉ những đơn hàng có tổng giá trị lớn hơn \$1000 bằng cách sử dụng HAVING như sau:

```
SELECT ordernumber,  
       sum(quantityOrdered) AS itemCount,  
       sum(priceEach * quantityOrdered) AS total  
FROM orderdetails  
GROUP BY ordernumber  
HAVING total > 1000
```

	ordernumber	itemCount	total
►	10100	151	10223.829999999998
	10101	142	10549.01
	10102	80	5494.78
	10103	541	50218.950000000004
	10104	443	40206.2
	10105	545	53959.21
	10106	675	52151.810000000005
	10107	229	22292.620000000003
	10108	561	51001.219999999994
	10109	212	25833.14
	10110	570	48425.69
	10111	217	16537.850000000002
	10112	52	7674.9400000000005
	10113	143	11044.300000000001
	10114	351	33383.140000000001
	10115	210	21665.980000000003
	10116	27	1627.56
	10117	402	44380.15

Chúng ta sử dụng bí danh cho cột `sum(priceEach * quantityOrdered)` là *total*, như vậy trong mệnh đề HAVING, chúng ta chỉ cần dùng *bí danh* đó thay vì gõ `sum(priceeach)` một lần nữa.

Có thể sử dụng một điều kiện kết hợp trong mệnh đề HAVING với các toán tử OR, AND.

Ví dụ: nếu muốn biết những đơn hàng có tổng giá trị lớn hơn \$ 1000 và có hơn 600 mặt hàng trong đó, có thể sử dụng truy vấn sau đây:

```

SELECT ordernumber,
       sum(quantityOrdered) AS itemCount,
       sum(priceeach) AS total
FROM orderdetails
GROUP BY ordernumber
HAVING total > 1000 AND itemCount > 600

```

	ordernumber	itemCount	total
►	10106	675	1427.2800000000002
	10126	617	1623.71
	10135	607	1494.86
	10165	670	1794.9399999999996
	10168	642	1472.5
	10204	619	1619.73
	10207	615	1560.08
	10212	612	1541.8300000000002
	10222	717	1389.51
	10262	605	1217.38
	10275	601	1455.4099999999999
	10310	619	1656.2600000000002
	10312	601	1494.1900000000003
	10316	623	1375.59

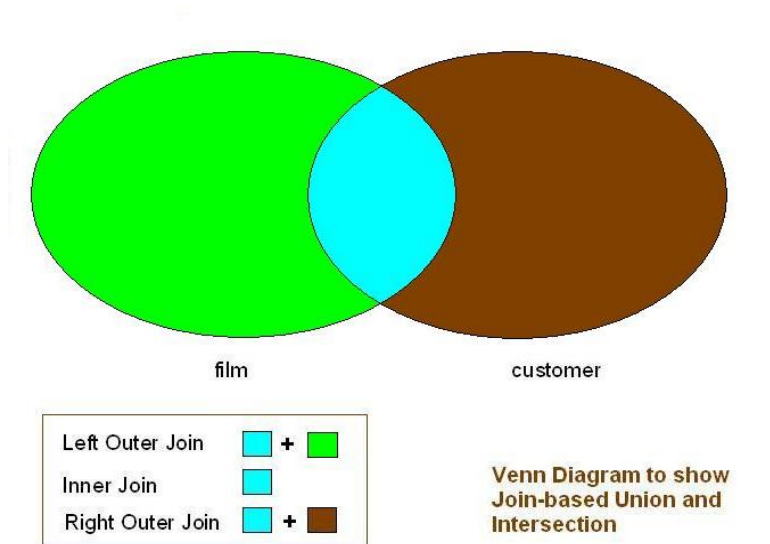
❖ Bài tập thực hành

- Đưa ra tên các thành phố và số lượng khách hàng tại từng thành phố.
- Đưa ra số lượng các đơn đặt hàng trong tháng 3/2005.
- Đưa ra số lượng các đơn đặt hàng trong từng tháng của năm 2005
- Đưa ra 10 mã đơn đặt hàng có giá trị lớn nhất.
- Đưa ra mã nhóm hàng và tổng số lượng hàng hoá còn trong kho của nhóm hàng đó.

Bài thực hành số 7

Các phép nối bảng dữ liệu

❖ **Nội dung chính:** Trong các bài thực hành trước, các truy vấn được thực hiện trên một bảng dữ liệu. Không ngạc nhiên khi rất nhiều truy vấn yêu cầu thông tin từ nhiều bảng dữ liệu khác nhau. Ví dụ muốn đưa ra thông tin khách hàng của các đơn hàng, cần kết hợp thông tin từ hai bảng dữ liệu là *customers* và *orders*. Kết hợp các bảng dữ liệu để tạo ra một bảng suy diễn được gọi là phép nối (*join*). Trong bài này, chúng ta sẽ làm quen với phép toán nối để truy vấn dữ liệu từ nhiều bảng : *INNER JOIN*, *LEFT JOIN*, *SELF JOIN*



1. PHÉP NỐI TRONG (INNER JOIN)

INNER JOIN hay còn gọi là phép nối trong, là một phần tùy chọn của câu lệnh SELECT. Nó xuất hiện liền ngay sau mệnh đề FROM. Trước khi sử dụng INNER JOIN, phải xác định rõ các tiêu chí sau đây:

- Trước tiên, cần phải xác định các bảng mà muốn liên kết với bảng chính. Bảng chính xuất hiện trong mệnh đề FROM. Bảng muốn nối với bảng chính phải xuất hiện sau từ khóa INNER JOIN. Về mặt lý thuyết, có thể nối một bảng với số

lượng không giới hạn các bảng khác, tuy nhiên, để có hiệu suất tốt hơn, nên hạn chế số lượng bảng tham gia phép nối dựa trên các điều kiện nối và khối lượng dữ liệu trong các bảng.

- Thứ hai, cần phải xác định điều kiện nối. Điều kiện nối xuất hiện sau từ khóa ON. Điều kiện nối chính là nguyên tắc để tìm được các bản ghi phù hợp trong các bảng và nối chúng lại với nhau.

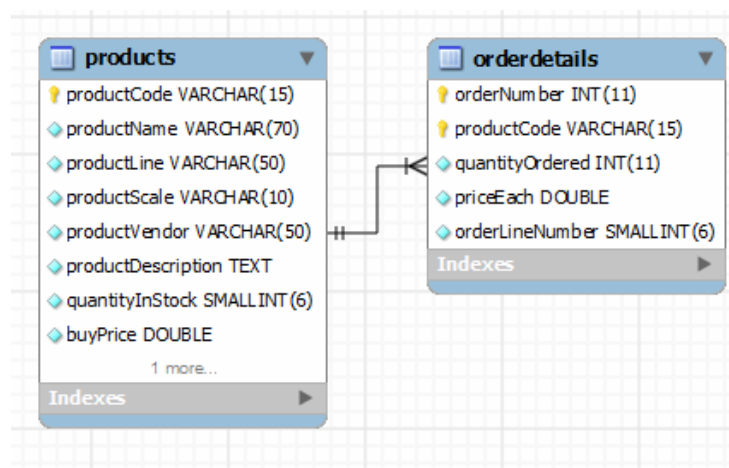
Cú pháp INNER JOIN như sau:

```
SELECT column_list
FROM table1
INNER JOIN table2 ON join_condition1
INNER JOIN table3 ON join_condition2
...
WHERE WHERE_conditions;
```

Ví dụ, nếu nối hai bảng A và B, INNER JOIN so sánh mỗi bản ghi của bảng A với mỗi bản ghi của bảng B để tìm tất cả các cặp bản ghi đáp ứng được điều kiện nối. Khi điều kiện nối được thoả mãn, giá trị cột cho mỗi cặp bản ghi phù hợp của bảng A và bảng B được kết hợp thành một bản ghi trong kết quả trả về.

Hạn chế sự trùng tên cột khi sử dụng INNER JOIN: Nếu nối nhiều bảng có cột với tên tương tự, phải chỉ rõ tên bảng có chứa cột dữ liệu định lấy để tránh lỗi cột không rõ ràng. Giả sử nếu bảng *tbl_A* và *tbl_B* có các cột tương tự *M*. Trong câu lệnh SELECT với INNER JOIN, phải tham chiếu tới cột *M* bằng cách sử dụng cú pháp như *tbl_A.M*.

Ví dụ: Hãy xem xét hai bảng *products* và *orderDetails*. Bảng *products* là bảng dữ liệu tổng thể lưu trữ tất cả các sản phẩm. Bất cứ khi nào một sản phẩm được bán ra, nó được lưu trữ trong bảng *OrderDetails* cùng với các thông tin khác. Liên kết giữa các bảng này là cột *productCode*



Ví dụ: muốn biết những sản phẩm đã được bán, có thể sử dụng *INNER JOIN* như sau:

```

SELECT products.productCode, products.productName,
orderDetails.orderNumber
FROM products
INNER JOIN orderDetails on products.productCode =
orderDetails.productCode;
  
```

productCode	productName	orderNumber
S18_1749	1917 Grand Touring Sedan	10100
S18_2248	1911 Ford Town Car	10100
S18_4409	1932 Alfa Romeo 8C2300 Spider Sport	10100
S24_3969	1936 Mercedes Benz 500k Roadster	10100
S18_2325	1932 Model A Ford J-Coupe	10101
S18_2795	1928 Mercedes-Benz SSK	10101
S24_1937	1939 Chevrolet Deluxe Coupe	10101
S24_2022	1938 Cadillac V-16 Presidential Limousine	10101
S18_1342	1937 Lincoln Berline	10102
S18_1367	1936 Mercedes-Benz 500K Special Roadster	10102
S10_1949	1952 Alpine Renault 1300	10103
S10_4962	1962 LanciaA Delta 16V	10103
S12_1666	1958 Setra Bus	10103
S18_1097	1940 Ford Pickup Truck	10103

INNER JOIN so sánh từng dòng trong bảng products và OrderDetails để tìm một cặp bản ghi có cùng productCode. Nếu một cặp bản ghi có cùng mã sản phẩm, khi đó tên sản phẩm và số thứ tự cũng sẽ được kết hợp thành một hàng để trả lại kết quả.

Bí danh (Alias): có thể tạo bí danh của bảng *tbl_A* là *A* và tham chiếu đến cột *M* là *A.M*, như vậy không mất công gõ lại tên bảng nữa. Ví dụ trên có thể viết lại như sau:

```
SELECT p.productCode, p.productName, o.orderNumber
FROM products p
INNER JOIN orderDetails o on p.productCode = o.productCode;
```

Lưu ý: Bên cạnh phép nối trong sử dụng mệnh đề INNER JOIN .. ON, có thể nối trong hai bảng bằng cách đưa điều kiện nối vào mệnh đề WHERE. Ví dụ trên có thể viết lại như sau:

```
SELECT p.productCode, p.productName, o.orderNumber
FROM products p, orderDetails o
WHERE p.productCode = o.productCode;
```

Chúng ta sẽ xem xét một số ví dụ khác sử dụng phép nối dưới đây:

Ví dụ: Bảng Employees là bảng lưu giữ thông tin về các nhân viên của công ty; bảng Customers là bảng lưu giữ thông tin của các khách hàng, trong đó có thông tin liên quan đến mã số của nhân viên chăm sóc khách hàng. Như vậy liên kết giữa hai bảng này được thực hiện thông qua cột employeeNumber của bảng Employees và cột salesRep employeeNumber của bảng Customers.

Để biết thông tin về khách hàng và tên nhân viên chăm sóc khách hàng đó, có thể viết truy vấn sử dụng INNER JOIN như sau:

```
SELECT customerName, firstname as EmployeeName
FROM customers C join employees E
on C.salesrepemployeenumber = e.employeenumber
```

Kết quả trả về như sau:

	customerName	EmployeeName
►	Atelier graphique	Gerard
	Signal Gift Stores	Leslie
	Australian Collectors, Co.	Andy
	La Rochelle Gifts	Gerard
	Baane Mini Imports	Barry
	Mini Gifts Distributors Ltd.	Leslie
	Blauer See Auto, Co.	Barry
	Mini Wheels Co.	Leslie
	Land of Toys Inc.	George
	Euro+ Shopping Channel	Gerard
	Volvo Model Replicas, Co	Barry
	Danish Wholesale Imports	Pamela
	Saveley & Henriot, Co.	Loui
	Dragon Souvenirs, Ltd.	Mami
	Muscle Machine Inc	Foon Yue
	Diecast Classics Inc.	Steve
	Technics Stores Inc.	Leslie

Ví dụ: Đưa ra thông tin về các dòng sản phẩm và tổng số hàng có trong dòng sản phẩm đó.

```
SELECT pl.productLine, pl.textDescription,
sum(quantityInStock)
FROM productlines pl JOIN products p ON pl.productLine
=p.productLine
GROUP BY pl.productLine;
```

Kết quả trả về như sau:

	productLine	textDescription	sum(quantityInStock)
►	Classic Cars	Attention car enthusiasts: Ma...	219183
	Motorcycles	Our motorcycles are state of t...	69401
	Planes	Unique, diecast airplane and ...	62287
	Ships	The perfect holiday or anniver...	26833
	Trains	Model trains are a rewarding ...	16696
	Trucks and Buses	The Truck and Bus models ar...	35851
	Vintage Cars	Our Vintage Car models realist...	124880

Ví dụ: Đưa ra thông tin về các sản phẩm và tổng giá trị đã đặt hàng cho sản phẩm, sắp xếp theo tổng giá trị tăng dần.

```
SELECT P.productCode,
       P.productName,
       SUM(priceEach * quantityOrdered) total
FROM orderdetails O
INNER JOIN products P ON O.productCode = P.productCode
GROUP BY productCode
ORDER BY total
```

Kết quả trả về như sau:

	productCode	productName	total
►	S24_1937	1939 Chevrolet Deluxe Coupe	28052.94
	S24_3969	1936 Mercedes Benz 500k Roadster	29763.39
	S24_2972	1982 Lamborghini Diablo	30972.869999999995
	S24_2840	1958 Chevy Corvette Limited Edition	31627.960000000003
	S32_2206	1982 Ducati 996 R	33268.76
	S24_2022	1938 Cadillac V-16 Presidential Limousine	38449.090000000004
	S50_1341	1930 Buick Marquette Phaeton	41599.24
	S24_1628	1966 Shelby Cobra 427 S/C	42015.539999999999
	S72_1253	Boeing X-32A JSF	42692.53
	S18_4668	1939 Cadillac Limousine	44037.839999999999
	S18_2248	1911 Ford Town Car	45306.770000000004
	S18_1367	1936 Mercedes-Benz 500K Special Roadster	46078.29
	S32_2509	1954 Greyhound Scenicruiser	46519.049999999998
	S72_3212	Pont Yacht	47550.399999999994

Bên cạnh phép nối hai bảng dữ liệu, ta có thể nối nhiều bảng dữ liệu trong cùng một câu lệnh SELECT.

Ví dụ: Đưa ra tên các khách hàng và tổng giá trị các đơn hàng của các khách hàng đó.

```
SELECT C.customerName, sum(OD.priceEach*OD.quantityOrdered)
as total
FROM customers C
INNER JOIN orders O on C.customerNumber = O.customerNumber
INNER JOIN orderdetails OD on O.orderNumber =
OD.orderNumber
GROUP BY C.customerName
```

Như trong ví dụ trên thông tin cần kết hợp từ ba bảng dữ liệu là customers, orders và orderdetails.

	customerName	total
►	Alpha Cognac	60483.359999999986
	Amica Models & Co.	82223.23
	Anna's Decorations, Ltd	137034.219999999994
	Atelier graphique	22314.36
	Australian Collectables, Ltd	55866.02
	Australian Collectors, Co.	180585.07
	Australian Gift Network, Co	55190.16
	Auto Associés & Cie.	58876.409999999996
	Auto Canal+ Pettit	86436.969999999999
	Auto-Moto Classics Inc.	21554.260000000002
	AV Stores, Co.	148410.090000000003
	Baane Mini Imports	104224.789999999996
	Bavarian Collectables Imports, Co.	31310.089999999997
	Blauer See Auto, Co.	75937.76
	Boards & Toys Co.	7918.6
	CAF Imports	46751.140000000001
	Cambridge Collectables Co.	32198.69

Ví dụ: Đưa ra các đơn hàng, tên các khách hàng và tổng giá trị của đơn hàng đó.

```
SELECT O.orderNumber,C.customerName,
sum(OD.priceEach*OD.quantityOrdered) as total
FROM customers C
INNER JOIN orders O on C.customerNumber = O.customerNumber
INNER JOIN orderdetails OD on O.orderNumber =
OD.orderNumber
GROUP BY O.orderNumber;
```

	orderNumber	customerName	total
▶	10100	Online Diecast Creations Co.	10223.829999999998
	10101	Blauer See Auto, Co.	10549.01
	10102	Vitachrome Inc.	5494.78
	10103	Baane Mini Imports	50218.950000000004
	10104	Euro+ Shopping Channel	40206.2
	10105	Danish Wholesale Imports	53959.21
	10106	Rovelli Gifts	52151.810000000005
	10107	Land of Toys Inc.	22292.620000000003
	10108	Cruz & Sons Co.	51001.219999999994
	10109	Motor Mint Distributors Inc.	25833.14
	10110	AV Stores, Co.	48425.69
	10111	Mini Wheels Co.	16537.850000000002
	10112	Volvo Model Replicas, Co	7674.9400000000005
	10113	Mini Gifts Distributors Ltd.	11044.300000000001
	10114	La Come D'abondance, Co.	33383.140000000001
	10115	Classic Legends Inc.	21665.980000000003
	10116	Royale Belge	1627.56

2. PHÉP NỐI TRÁI (LEFT JOIN)

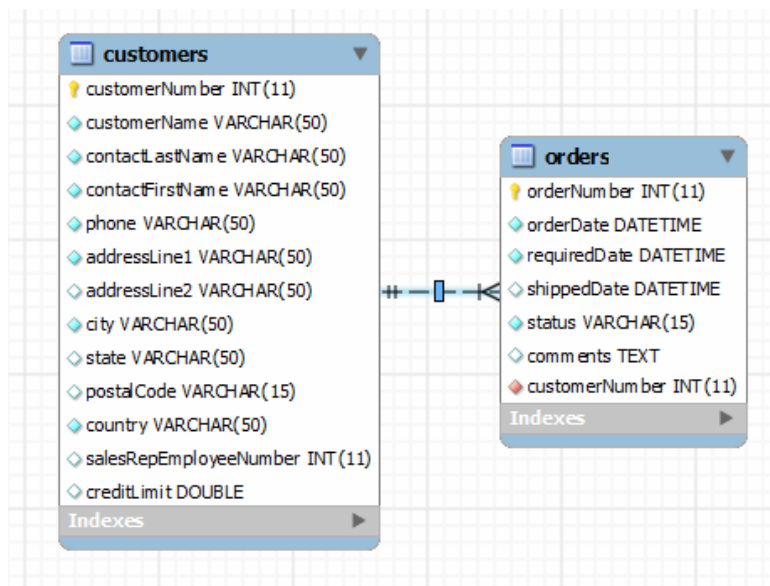
LEFT JOIN cũng là một tùy chọn của câu lệnh SELECT cho phép lấy thêm dữ liệu từ các bảng khác. LEFT JOIN bao gồm các từ khóa LEFT JOIN, tiếp theo là bảng thứ hai muốn thực hiện nối. Yếu tố tiếp theo là từ khóa ON và theo sau bởi các điều kiện nối.

Mệnh đề LEFT JOIN sẽ được thực hiện như sau: khi một hàng từ bảng bên trái phù hợp với một hàng từ bảng bên phải dựa trên điều kiện nối, nội dung của hàng đó sẽ được lựa chọn như một dòng trong kết quả đầu ra. Khi một hàng trong bảng bên trái không tìm được hàng nào phù hợp trong bảng nối, nó vẫn được xuất hiện trong kết quả đầu ra, nhưng kết hợp với một hàng "giả" từ bảng bên phải với giá trị NULL cho tất cả các cột.

Tóm lại, LEFT JOIN cho phép chọn tất cả các hàng từ bảng bên trái ngay cả khi không có bản ghi nào phù hợp với nó trong bảng bên phải.

Ví dụ: sử dụng LEFT JOIN

Chúng ta hãy xét vào hai bảng customers và orders. Nếu muốn biết một khách hàng với hoá đơn nào đó của họ và tình trạng hoá đơn đó thế nào, có thể sử dụng MySQL LEFT JOIN như sau:



```
SELECT c.customerNumber, customerName,orderNUmber, o.status
FROM customers c
LEFT JOIN orders o ON c.customerNumber = o.customerNumber;
```


	customerNumber	customerName	orderNumber	status
	124	Mini Gifts Distributors L...	10371	Shipped
	124	Mini Gifts Distributors L...	10382	Shipped
	124	Mini Gifts Distributors L...	10385	Shipped
	124	Mini Gifts Distributors L...	10390	Shipped
	124	Mini Gifts Distributors L...	10396	Shipped
	124	Mini Gifts Distributors L...	10421	In Process
	125	Havel & Zbyszek Co	NULL	NULL
	128	Blauer See Auto, Co.	10101	Shipped
	128	Blauer See Auto, Co.	10230	Shipped
	128	Blauer See Auto, Co.	10300	Shipped
	128	Blauer See Auto, Co.	10323	Shipped

Ở bảng kết quả trên, có thể nhìn thấy tất cả các khách hàng được liệt kê. Tuy nhiên, có những bản ghi có thông tin khách hàng nhưng tất cả các thông tin về đơn hàng là NULL. Điều này có nghĩa là những khách hàng này không có bất kỳ một đơn đặt hàng nào được lưu trong cơ sở dữ liệu của chúng ta.

LEFT JOIN rất hữu ích khi muốn tìm các bản ghi trong bảng bên trái mà không phù hợp với bất kỳ một bản ghi nào trong bảng bên phải. có thể thực hiện điều này bằng cách thêm một mệnh đề WHERE để lựa chọn các hàng chỉ có giá trị NULL trong một cột ở bảng bên phải. Vì vậy, để tìm thấy tất cả các khách hàng không có bất kỳ đơn đặt hàng nào trong cơ sở dữ liệu của chúng ta, có thể sử dụng LEFT JOIN như sau:

```
SELECT c.customerNumber, customerName,orderNumber, o.status
FROM customers c
LEFT JOIN orders o ON c.customerNumber = o.customerNumber
WHERE orderNumber is NULL
```

Kết quả trả về như sau:

	customerNumber	customerName	orderNumber	status
▶	125	Havel & Zbyszek Co	NULL	NULL
	168	American Souvenirs Inc	NULL	NULL
	169	Porto Imports Co.	NULL	NULL
	206	Asian Shopping Network, Co	NULL	NULL
	223	Natürlich Autos	NULL	NULL
	237	ANG Resellers	NULL	NULL
	247	Messner Shopping Network	NULL	NULL
	273	Franken Gifts, Co	NULL	NULL
	293	BG&E Collectables	NULL	NULL
	303	Schuyler Imports	NULL	NULL

Như vậy, truy vấn chỉ trả về các khách hàng mà không có bất kỳ đơn hàng nào nhờ vào các giá trị NULL.

Tương tự như vậy, để tìm ra những nhân viên không làm nhiệm vụ chăm sóc khách hàng, bước đầu, thực hiện truy vấn như sau:

```

Select * from employees e
left join customers c
on e.employeeNumber=c.salesrepEmployeeNumber

```

	employeeNumber	lastName	firstName	extension	email	officeCode	reportsTo	customerNumber
▶	1002	Murphy	Diane	x5800	dmurphy@classicmodelcars.com	1	NULL	NULL
	1056	Patterson	Mary	x4611	mpatterso@classicmodelcars.com	1	1002	NULL
	1076	Firelli	Jeff	x9273	jfirelli@classicmodelcars.com	1	1002	NULL
	1088	Patterson	William	x4871	wpatterson@classicmodelcars.com	6	1056	NULL
	1102	Bondur	Gerard	x5408	gbondur@classicmodelcars.com	4	1056	NULL
	1143	Bow	Anthony	x5428	abow@classicmodelcars.com	1	1056	NULL
	1165	Jennings	Leslie	x3291	ljennings@classicmodelcars.com	1	1143	124
	1165	Jennings	Leslie	x3291	ljennings@classicmodelcars.com	1	1143	129
	1165	Jennings	Leslie	x3291	ljennings@classicmodelcars.com	1	1143	161
	1165	Jennings	Leslie	x3291	ljennings@classicmodelcars.com	1	1143	321
	1165	Jennings	Leslie	x3291	ljennings@classicmodelcars.com	1	1143	450
	1165	Jennings	Leslie	x3291	ljennings@classicmodelcars.com	1	1143	487
	1166	Thompson	Leslie	x4065	lthompson@classicmodelcars.com	1	1143	112
	1166	Thompson	Leslie	x4065	lthompson@classicmodelcars.com	1	1143	205
	1166	Thompson	Leslie	x4065	lthompson@classicmodelcars.com	1	1143	219
	1166	Thompson	Leslie	x4065	lthompson@classicmodelcars.com	1	1143	239
	1166	Thompson	Leslie	x4065	lthompson@classicmodelcars.com	1	1143	347
	1166	Thompson	Leslie	x4065	lthompson@classicmodelcars.com	1	1143	475

Sau đó lọc ra những bản ghi nhận giá trị null tại cột customerNumber, đó chính là kết quả của truy vấn.

```

Select * from employees e
left join customers c
on e.employeeNumber=c.salesrepemployeeNumber
where customerNumber is null

```

	employeeNumber	lastName	firstName	extension	email	officeCode	reportsTo	customerNumber
▶	1002	Murphy	Diane	x5800	dmurphy@classicmodelcars.com	1	NULL	NULL
	1056	Patterson	Mary	x4611	mpatterso@classicmodelcars.com	1	1002	NULL
	1076	Firelli	Jeff	x9273	jfirelli@classicmodelcars.com	1	1002	NULL
	1088	Patterson	William	x4871	wpatterson@classicmodelcars.com	6	1056	NULL
	1102	Bondur	Gerard	x5408	gbondur@classicmodelcars.com	4	1056	NULL
	1143	Bow	Anthony	x5428	abow@classicmodelcars.com	1	1056	NULL
	1619	King	Tom	x103	tking@classicmodelcars.com	6	1088	NULL
	1625	Kato	Yoshimi	x102	ykato@classicmodelcars.com	5	1621	NULL

3. PHÉP TỰ NỐI (Self Join)

Một phép tự nối là một kiểu nối trong đó một bảng được nối với chính nó, cụ thể khi một bảng có một khóa ngoài tham chiếu tới khóa chính của nó.

Ví dụ: Bảng *employees* có một khóa ngoài là *reportsTo* tham chiếu tới khóa chính *employeeNumber* của chính bảng *employees*.

Cần thiết phải sử dụng bí danh cho mỗi bản sao của bảng đó để tránh nhập nhằng

```

SELECT concat (e1.lastName , " ", e1.firstName) as fullname,
e1.email, concat (e2.lastName , " ", e2.firstName) as
manager, e2.email
FROM employees e1, employees e2
WHERE e1.reportsTo = e2.employeeNumber;

```

Kết quả trả về như sau:

	fullname	email	manager	email
►	Patterson Mary	mpatterso@classicmodelcars.com	Murphy Diane	dmurphy@classicmodelcars.com
	Firelli Jeff	jfirelli@classicmodelcars.com	Murphy Diane	dmurphy@classicmodelcars.com
	Patterson William	wpatterson@classicmodelcars.com	Patterson Mary	mpatterso@classicmodelcars.com
	Bondur Gerard	gbondur@classicmodelcars.com	Patterson Mary	mpatterso@classicmodelcars.com
	Bow Anthony	abow@classicmodelcars.com	Patterson Mary	mpatterso@classicmodelcars.com
	Jennings Leslie	ljennings@classicmodelcars.com	Bow Anthony	abow@classicmodelcars.com
	Thompson Leslie	lthompson@classicmodelcars.com	Bow Anthony	abow@classicmodelcars.com
	Firelli Julie	jfirelli@classicmodelcars.com	Bow Anthony	abow@classicmodelcars.com
	Patterson Steve	spatterson@classicmodelcars.com	Bow Anthony	abow@classicmodelcars.com
	Tseng Foon Yue	ftseng@classicmodelcars.com	Bow Anthony	abow@classicmodelcars.com
	Vanauf George	gvanauf@classicmodelcars.com	Bow Anthony	abow@classicmodelcars.com
	Bondur Loui	lbondur@classicmodelcars.com	Bondur Gerard	gbondur@classicmodelcars.com
	Hernandez Ger...	ghemande@classicmodelcars.com	Bondur Gerard	gbondur@classicmodelcars.com
	Castillo Pamela	pcastillo@classicmodelcars.com	Bondur Gerard	gbondur@classicmodelcars.com
	Bott Larry	lbott@classicmodelcars.com	Bondur Gerard	gbondur@classicmodelcars.com

❖ Bài tập thực hành:

- Đưa ra thông tin về các nhân viên và tên văn phòng nơi họ làm việc.
- Đưa ra thông tin về tên khách hàng và tên các sản phẩm họ đã mua.
- Đưa ra thông tin về các mặt hàng chưa có ai đặt mua.
- Đưa ra ~~thông tin về~~ các đơn hàng trong tháng 3/2005 (gồm orderDate, requiredDate, Status) và tổng giá trị của mỗi đơn hàng .
- Đưa ra thông tin về các dòng sản phẩm và số lượng sản phẩm của dòng sản phẩm đó. Sắp xếp theo thứ tự số lượng giảm dần.

Bài thực hành số 8

Truy vấn con (Subquery)

- ❖ **Nội dung chính:** Khái niệm và sử dụng truy vấn con, truy vấn con tương quan và không tương quan.

1. Khái niệm truy vấn con

Để kết hợp các bảng dữ liệu với nhau, ngoài các phép nối và các toán tử tập hợp, SQL cung cấp một cách khác để trả lại dữ liệu từ nhiều bảng gọi là truy vấn con (*subquery*). Khi một câu lệnh SELECT được sử dụng trong một câu lệnh khác, câu lệnh SELECT bên trong được gọi là truy vấn con (subquery), cách gọi khác là truy vấn lồng (nested query), truy vấn trong (inner query). Cơ bản một truy vấn con có thể được sử dụng ở bất cứ nơi đâu mà một biểu thức có thể được sử dụng.

Ví dụ: Đưa ra các đơn hàng gần đây nhất

```
SELECT * FROM orders
WHERE orderDate = (SELECT MAX(orderDate) FROM orders)
```

Truy vấn con `SELECT MAX(orderDate) FROM orders` trả lại ngày gần đây nhất trong các đơn hàng và giá trị này sẽ được sử dụng trong mệnh đề WHERE của truy vấn ngoài. Kết hợp hai truy vấn trên sẽ trả lại danh sách các đơn hàng của ngày gần đây nhất.

	orderNumber	orderDate	requiredDate	shippedDate	status	comments
▶	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

Truy vấn con được chia làm hai loại: truy vấn con không tương quan (non-correlated) và truy vấn con có tương quan (correlated)

2. Truy vấn con không tương quan

Một truy vấn con không tương quan là truy vấn con độc lập với truy vấn bên ngoài. Truy vấn con không tương quan được thi hành đầu tiên và một lần duy nhất cho toàn

bộ câu lệnh. Kết quả của truy vấn con được điền vào truy vấn bên ngoài, và cuối cùng thì hành truy vấn bên ngoài.

Ví dụ: đưa các sản phẩm không có mặt trong bất kỳ một đơn hàng nào. Truy vấn con bên trong sẽ trả về các mã sản phẩm có trong bảng orderdetails. Truy vấn bên ngoài sẽ trả về các sản phẩm có mã không trong danh sách các mã sản phẩm đó.

```
SELECT *
FROM products
WHERE productCode not in
    (SELECT productCode
     FROM orderdetails
    )
```

	productCode	productName	productLine	productScale	productVendor
►	S18_3233	1985 Toyota Supra	Classic Cars	1:18	Highway 66 Mini Classics

Ví dụ: đưa ra các sản phẩm có mặt trong các đơn hàng

```
SELECT * FROM products
WHERE productCode in
    (SELECT productCode
     FROM orderdetails
    )
```

	productCode	productName	productLine	productScale	productVendor	productDescription
►	S10_1678	1969 Harley Davidson Ultimate Chopper	Motorcycles	1:10	Min Lin Diecast	This replica features w
	S10_1949	1952 Alpine Renault 1300	Classic Cars	1:10	Classic Metal Creations	Tumble front wheels;
	S10_2016	1996 Moto Guzzi 1100i	Motorcycles	1:10	Highway 66 Mini Classics	Official Moto Guzzi log
	S10_4698	2003 Harley-Davidson Eagle Drag Bike	Motorcycles	1:10	Red Start Diecast	Model features, official
	S10_4757	1972 Alfa Romeo GTA	Classic Cars	1:10	Motor City Art Classics	Features include: Tum
	S10_4962	1962 Lancia Delta 16V	Classic Cars	1:10	Second Gear Diecast	Features include: Tum
	S12_1099	1968 Ford Mustang	Classic Cars	1:12	Autoart Studio Design	Hood, doors and trunk
	S12_1108	2001 Ferrari Enzo	Classic Cars	1:12	Second Gear Diecast	Tumble front wheels;
	S12_1666	1958 Setra Bus	Trucks and Buses	1:12	Welly Diecast Productions	Model features 30 win
	S12_2823	2002 Suzuki XREO	Motorcycles	1:12	Unimax Art Galleries	Official logos and insig
	S12_3148	1969 Corvair Monza	Classic Cars	1:18	Welly Diecast Productions	1:18 scale die-cast ab
	S12_3380	1968 Dodge Charger	Classic Cars	1:12	Welly Diecast Productions	1:12 scale model of a
	S12_3891	1969 Ford Falcon	Classic Cars	1:12	Second Gear Diecast	Tumble front wheels;
	S12_3999	1978 BMW 1100CC	Classic Cars	1:12	Second Gear Diecast	1:12 scale model of a

3. Truy vấn con tương quan

Truy vấn con tương quan không độc lập với truy vấn bên ngoài. Một truy vấn con tương quan là một truy vấn con sử dụng các giá trị từ truy vấn bên ngoài trong mệnh đề WHERE của nó. Quá trình thực hiện như sau: các truy vấn bên ngoài được thực hiện trước tiên và sau đó thì hành truy vấn con bên trong cho mỗi dòng kết quả của truy vấn bên ngoài.

Ví dụ: đưa ra các sản phẩm có số lượng trong kho lớn hơn trung bình số lượng trong kho của các sản phẩm cùng loại.

```
SELECT * FROM products p
WHERE quantityInStock >
      (SELECT avg(quantityInStock)
       FROM products
       WHERE productLine = p.productLine
      )
```

	productCode	productName	productLine	productScale	productVendor	productDescription
►	S10_1678	1969 Harley Davidson Ultimate Chopper	Motorcycles	1:10	Min Lin Diecast	This replica features w
	S10_1949	1952 Alpine Renault 1300	Classic Cars	1:10	Classic Metal Creations	Turnable front wheels
	S10_2016	1996 Moto Guzzi 1100i	Motorcycles	1:10	Highway 66 Mini Classics	Official Moto Guzzi log
	S10_4698	2003 Harley-Davidson Eagle Drag Bike	Motorcycles	1:10	Red Start Diecast	Model features, offici
	S10_4962	1962 Lancia Delta 16V	Classic Cars	1:10	Second Gear Diecast	Features include: Turr
	S12_2823	2002 Suzuki XREO	Motorcycles	1:12	Unimax Art Galleries	Official logos and insig
	S12_3148	1969 Corvair Monza	Classic Cars	1:18	Welly Diecast Productions	1:18 scale die-cast ab
	S12_3380	1968 Dodge Charger	Classic Cars	1:12	Welly Diecast Productions	1:12 scale model of a
	S12_4473	1957 Chevy Pickup	Trucks and Buses	1:12	Exoto Designs	1:12 scale die-cast ab
	S12_4675	1969 Dodge Charger	Classic Cars	1:12	Welly Diecast Productions	Detailed model of the

Quá trình thực hiện truy vấn như sau: với mỗi dòng sản phẩm của truy vấn bên ngoài, câu lệnh truy vấn bên trong sẽ tìm ra số lượng sản phẩm trung bình của sản phẩm cùng loại với sản phẩm đó và kết quả của truy vấn con sẽ được đưa vào mệnh đề WHERE để kiểm tra.

Ví dụ: đưa ra các sản phẩm có mặt trong các đơn hàng, cách viết dưới đây là một cách khác của ví dụ ở phần trước. Sử dụng toán tử EXISTS để kiểm tra sự tồn tại.

```
SELECT * FROM products as p
```

```

WHERE exists
      (SELECT productCode
FROM orderdetails
WHERE productCode = p.productCode)

```

	productCode	productName	productLine	productScale	productVendor	productDescription	quantityInStock
	S10_1678	1969 Harley Da...	Motorcycles	1:10	Min Lin Diecast	This replica features ...	7933
	S10_1949	1952 Alpine Ren...	Classic Cars	1:10	Classic Metal Cre...	Tumble front wheels...	7305
	S10_2016	1996 Moto Guzz...	Motorcycles	1:10	Highway 66 Mini ...	Official Moto Guzzi lo...	6625
	S10_4698	2003 Harley-Da...	Motorcycles	1:10	Red Start Diecast	Model features, officia...	5582
	S10_4757	1972 Alfa Rome...	Classic Cars	1:10	Motor City Art Cla...	Features include: Tur...	3252
	S10_4962	1962 LanciaA D...	Classic Cars	1:10	Second Gear Die...	Features include: Tur...	6791
	S12_1099	1968 Ford Must...	Classic Cars	1:12	Autoart Studio De...	Hood, doors and trun...	68
	S12_1108	2001 Ferrari Enzo	Classic Cars	1:12	Second Gear Die...	Tumble front wheels...	3619
	S12_1666	1958 Setra Bus	Trucks and Bu...	1:12	Welly Diecast Pro...	Model features 30 win...	1579
	S12_2823	2002 Suzuki XR...	Motorcycles	1:12	Unimax Art Galleries	Official logos and insi...	9997
	S12_3148	1969 Corvair Mo...	Classic Cars	1:18	Welly Diecast Pro...	1:18 scale die-cast ab...	6906
	S12_3380	1968 Dodge Ch...	Classic Cars	1:12	Welly Diecast Pro...	1:12 scale model of a ...	9123
	S12_3891	1969 Ford Falcon	Classic Cars	1:12	Second Gear Die...	Tumble front wheels...	1049
	S12_3990	1970 Plymouth ...	Classic Cars	1:12	Studio M Art Mod...	Very detailed 1970 Pl...	5663

4. Sử dụng truy vấn con

Ngoài sử dụng truy vấn con trong mệnh đề WHERE, truy vấn con còn có thể được sử dụng trong danh sách các cột của câu lệnh SELECT hoặc trong mệnh đề FROM.

Ví dụ: với mỗi dòng đơn hàng, đưa vào thêm tên của sản phẩm.

```

SELECT orderNumber, quantityOrdered,
      (SELECT productName FROM products WHERE productCode =
      o.productCode) as productName
FROM orderdetails o

```


	orderNumber	quantityOrdered	productName
▶	10100	30	1917 Grand Touring Sedan
	10100	50	1911 Ford Town Car
	10100	22	1932 Alfa Romeo 8C2300 Spider Sport
	10100	49	1936 Mercedes Benz 500k Roadster
	10101	25	1932 Model A Ford J-Coupe
	10101	26	1928 Mercedes-Benz 55K
	10101	45	1939 Chevrolet Deluxe Coupe
	10101	46	1938 Cadillac V-16 Presidential Limousine
	10102	39	1937 Lincoln Berline
	10102	41	1936 Mercedes-Benz 500K Special Roadster

Trong ví dụ trên tên của sản phẩm là kết quả của truy vấn con trên bảng *products*

Ví dụ: với mỗi sản phẩm, đưa kèm thêm tổng số lượng sản phẩm đó đã được đặt hàng

```
SELECT productName,
       (SELECT sum(quantityOrdered) FROM orderdetails WHERE
        productCode = p.productCode) as totalQuantityOrderd
FROM products as p
ORDER BY totalQuantityOrderd desc
```

	productName	totalQuantityOrderd
▶	1992 Ferrari 360 Spider red	1808
	1937 Lincoln Berline	1111
	American Airlines: MD-11S	1085
	1941 Chevrolet Special Deluxe Cabriolet	1076
	1930 Buick Marquette Phaeton	1074
	1940s Ford truck	1061
	1969 Harley Davidson Ultimate Chopper	1057
	1957 Chevy Pickup	1056
	1964 Mercedes Tour Bus	1053
	1956 Porsche 356A Coupe	1052
	Corsair F4U (Bird Cage)	1051
	F/A 18 Homet 1/72	1047
	1980s Black Hawk Helicopter	1040
	1913 Ford Model T Speedster	1038
	1997 BMW R 1100 S	1033

Trong ví dụ trên giá trị tổng số lượng được đặt là kết quả của truy vấn từ bảng *orderDetails*

Ví dụ trên có thể viết lại bằng cách coi kết quả của truy vấn con như một bảng dữ liệu, sau đó nối bảng *products* với bảng kết quả này.

```
SELECT productName, totalQuantityOrderd
FROM products,
(SELECT productCode,sum(quantityOrdered) as
totalQuantityOrderd FROM orderdetails group by
productCode) AS productOrder
WHERE products.productCode = productOrder.productCode
```

Kết quả của truy vấn cho kết quả tương tự như truy vấn trước

▶ 1992 Ferrari 360 Spider red	1808
1937 Lincoln Bertine	1111
American Airlines: MD-11S	1085
1941 Chevrolet Special Deluxe Cabriolet	1076
1930 Buick Marquette Phaeton	1074
1940s Ford truck	1061
1969 Harley Davidson Ultimate Chopper	1057
1957 Chevy Pickup	1056
1964 Mercedes Tour Bus	1053
1956 Porsche 356A Coupe	1052
Corsair F4U (Bird Cage)	1051
F/A 18 Homet 1/72	1047
1980s Black Hawk Helicopter	1040
1913 Ford Model T Speedster	1038
1997 BMW R 1100 S	1033

❖ Bài tập thực hành

1. Sử dụng truy vấn con đưa ra các sản phẩm có đơn đặt hàng trong tháng 3/2005.
2. Tương tự như câu hỏi 1 nhưng dùng phép nối bảng thay vì sử dụng truy vấn con.
3. Sử dụng truy vấn con đưa ra các thông tin về các đơn hàng trong tháng gần nhất (sử dụng thông tin từ bảng orders).
4. Sử dụng truy vấn con đưa ra thông tin về các đơn hàng và tổng giá trị đơn hàng (sử dụng thông tin từ bảng orders và orderdetails).

5. Cũng như câu hỏi 4, nhưng sử dụng phép nối bảng thay vì sử dụng truy vấn con.

5-6. Với mỗi khách hàng, đưa ra tổng số tiền hàng, và tổng số tiền họ đã thanh toán

Formatted: Font: Bold

Bài thực hành số 9

Thêm, sửa, xóa dữ liệu trong bảng

❖ **Nội dung:** Các câu lệnh cập nhật dữ liệu

- Lệnh INSERT
- Câu lệnh UPDATE
- Câu lệnh DELETE

1. Câu lệnh INSERT

Câu lệnh INSERT cho phép thêm các dòng dữ liệu vào một bảng xác định. Có hai biến thể của lệnh INSERT: cách thứ nhất là thêm một dòng giá trị, cách thứ hai là thêm một tập các dòng trả về từ một câu lệnh SELECT.

Thêm một dòng giá trị

```
INSERT INTO table_name
[(column_name, ...)]
VALUES ((expression | DEFAULT), ...), (...), ...
```

INSERT tạo một dòng mới trong bảng *<table_name>*. Dòng mới chứa các giá trị xác định bởi các biểu thức trong danh sách VALUES. Nếu *column_name* không được đưa vào, thì trình tự các cột trong bảng *<table_name>* được sử dụng. Nếu *column_name* được đưa, theo cách này, dòng dữ liệu mới được thêm vào bảng bằng cách xác định tên cột và dữ liệu cho mỗi cột.

Ví dụ: thêm một bản ghi vào bảng *offices*

```
INSERT INTO classicmodels.offices
  (officeCode,
   city,
   phone,
   addressLine1,
```

```

addressLine2,
state,
country,
postalCode,
territory
)
VALUES
('8',
'Boston',
'+1 215 837 0825',
'1550 dummy street',
NULL,
'MA',
'USA',
'02107',
'NA'
)

```

Nếu giá trị chưa xác định có thể sử dụng từ khóa NULL. Sử dụng giá trị ngầm định bằng từ khóa DEFAULT.

Có thể kiểm tra kết quả của lệnh trên bằng câu lệnh truy vấn:

```
SELECT * FROM classicmodels.offices
```

	officeCode	city	phone	addressLine1	addressLine2	state	country	postalCode	territory
▶	1	San Francisco	+1 650 219 4782	100 Market Street	Suite 300	CA	USA	94080	NA
	2	Boston	+1 215 837 0825	1550 Court Place	Suite 102	MA	USA	02107	NA
	3	NYC	+1 212 555 3000	523 East 53rd Street	apt. 5A	NY	USA	10022	NA
	4	Paris	+33 14 723 4404	43 Rue Jouffroy D'abbans	NULL	NULL	France	75017	EMEA
	5	Tokyo	+81 33 224 5000	4-1 Koicho	NULL	Chiyoda-Ku	Japan	102-8578	Japan
	6	Sydney	+61 2 9264 2451	5-11 Wentworth Avenue	Floor #2	NULL	Australia	NSW 2010	APAC
	7	London	+44 20 7877 2041	25 Old Broad Street	Level 7	NULL	UK	EC2N 1HN	EMEA
	8	Boston	+1 215 837 0825	1550 dummy street	NULL	MA	USA	02107	NA

Kết quả là một dòng dữ liệu mới được ghi vào cuối bảng dữ liệu

Chú ý: Nếu không xác định tên các cột, khi trật tự của các cột thay đổi, SQL có thể đưa giá trị vào sai vị trí. Do đó cách tốt để tránh điều này là xác định tên cột đi kèm với dữ liệu khi thêm dữ liệu vào bảng.

Thêm nhiều dòng với lệnh SELECT

Ngoài ra thay vì cung cấp dữ liệu trực tiếp, có thể chọn từ các bảng khác sử dụng câu lệnh SELECT.

```
INSERT INTO table_name
    [ (column_name, ...) ]
    <SELECT statement>;
```

Không giống với cách trước, cách này cho phép tạo nhiều dòng dữ liệu với. Danh sách các cột kết quả của lệnh SELECT phải trùng với danh sách các cột của bảng. Cũng giống như cách trước, các cột không xác định sẽ được gán giá trị ngầm ngấm của cột.

Ví dụ: tạo một bảng tạm và thêm vào tất cả các offices tại US

```
INSERT INTO temp_table
SELECT *
FROM classicmodels.offices
WHERE country = 'USA'
```

Có thể kiểm tra kết quả của lệnh trên bằng câu lệnh truy vấn:

```
SELECT * FROM classicmodels.temp_table
```

	officeCode	city	phone	addressLine1	addressLine2	state	country	postalCode	territory
►	1	San Francisco	+1 650 219 4782	100 Market Street	Suite 300	CA	USA	94080	NA
	2	Boston	+1 215 837 0825	1550 Court Place	Suite 102	MA	USA	02107	NA
	3	NYC	+1 212 555 3000	523 East 53rd Street	apt. 5A	NY	USA	10022	NA
	8	Boston	+1 215 837 0825	1550 dummy street	NULL	MA	USA	02107	NA

2. Câu lệnh UPDATE

Câu lệnh UPDATE được sử dụng để cập nhật dữ liệu đã tồn tại trong các bảng của CSDL. Câu lệnh có thể dùng để thay đổi các giá trị của một dòng, một nhóm các dòng hoặc thậm chí tất cả các dòng trong một bảng. Cấu trúc của câu lệnh UPDATE như sau:

```
UPDATE table_name [, table_name...]  
    SET column_name1=expr1  
        [, column_name2=expr2 ...]  
[WHERE condition]
```

- Sau từ khóa UPDATE là tên bảng muốn thay đổi dữ liệu. Mệnh đề SET xác định cột thay đổi và giá trị thay đổi. Giá trị thay đổi có thể là giá trị cố định, biểu thức hoặc thậm chí một truy vấn con.
- Mệnh đề WHERE xác định các dòng của bảng sẽ được cập nhật. Nếu mệnh đề WHERE bị bỏ qua, tất cả các dòng của bảng sẽ bị cập nhật.
- *Mệnh đề WHERE rất quan trọng, không nên bị bỏ qua. Nếu chỉ muốn thay đổi một dòng của một bảng, nhưng quên mệnh đề WHERE sẽ cập nhật toàn bộ bảng.*
- Nếu một câu lệnh UPDATE vi phạm bất cứ ràng buộc toàn vẹn nào, MySQL sẽ không thực hiện cập nhật và đưa ra thông báo lỗi

Ví dụ: Trong bảng *employees*, nếu muốn cập nhật email của Diane Murphy với employeeNumber là 1002 thành diane-murphy@classicmodelcars.com,

Thực hiện câu truy vấn sau:

```
SELECT firstname,  
        lastname,  
        email  
FROM employees  
WHERE employeeNumber = 1002
```

	firstname	lastname	email
▶	Diane	Murphy	dmurphy@classicmodelcars.com

Kết quả đã cập nhật email mới diane-murphy@classicmodelcars.com

```
UPDATE employees
SET email = 'diane-murphy @classicmodelcars.com'
WHERE employeeNumber = 1002
```

Thực hiện câu truy vấn SELECT lại, sẽ thấy email thay đổi giá trị mới:

	firstname	lastname	email
▶	Diane	Murphy	diane-murphy @classicmodelcars.com

3. Câu lệnh DELETE

Để xóa các dòng dữ liệu của một bảng CSDL, sử dụng câu lệnh DELETE.

Cấu trúc lệnh DELETE như sau:

```
DELETE FROM table_name
[WHERE conditions]
```

- Sau DELETE FROM là tên bảng muốn xóa các bản ghi. Mệnh đề WHERE xác định điều kiện để giới hạn các dòng muốn loại bỏ. Nếu một bản ghi thỏa mãn điều kiện WHERE sẽ bị loại bỏ khỏi bảng CSDL.
- Nếu mệnh đề WHERE bị bỏ qua trong câu lệnh DELETE, tất cả các dòng của bảng sẽ bị xóa. Để giảm sự nguy hiểm của các câu lệnh như DELETE hoặc UPDATE, nên luôn luôn kiểm tra điều kiện WHERE trong một câu lệnh SELECT trước khi thực hiện lệnh DELETE hoặc UPDATE.

Ví dụ: Xóa tất cả các nhân viên trong văn phòng có mã *officeNumber* là 6, thực hiện câu truy vấn sau:


```
DELETE
FROM employees
WHERE officeCode = 6
```

Thực hiện lại câu lệnh truy vấn trên bảng *employees*. Trong bảng không còn các dòng có *officeCode* = 6

	employeeNumber	lastName	firstName	extension	email	officeCode	reportsTo	jobTitle
▶	1002	Murphy	Diane	x5800	diane-murphy@classicmodelcars.com	1	NONE	President
	1056	Patterson	Mary	x4611	mpatterso@classicmodelcars.com	1	1002	VP Sales
	1076	Firelli	Jeff	x9273	jfirelli@classicmodelcars.com	1	1002	VP Marketing
	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	Firelli	Julie	x2173	jfirelli@classicmodelcars.com	2	1143	Sales Rep

Chú ý: Nếu loại bỏ điều kiện WHERE

```
DELETE FROM employees
```

Sẽ xóa tất cả các dòng của bảng *employees*. Do đó cần chú ý điều kiện trong mệnh đề WHERE khi thực hiện lệnh DELETE.

MySQL cũng hỗ trợ xóa các bản ghi từ nhiều bảng khác nhau.

Ví dụ: xóa tất cả các nhân viên (employee) làm việc cho văn phòng có mã *officecode* 1 và cũng xóa cả văn phòng đó.

```
DELETE employees,offices
FROM employees,offices
WHERE employees.officeCode = offices.officeCode AND
      offices.officeCode = 1
```

Sau khi thực hiện lệnh xóa dữ liệu trên, kiểm tra lại các bảng dữ liệu
Bảng *employees* không còn các dòng nhân viên có *officeCode* = 1

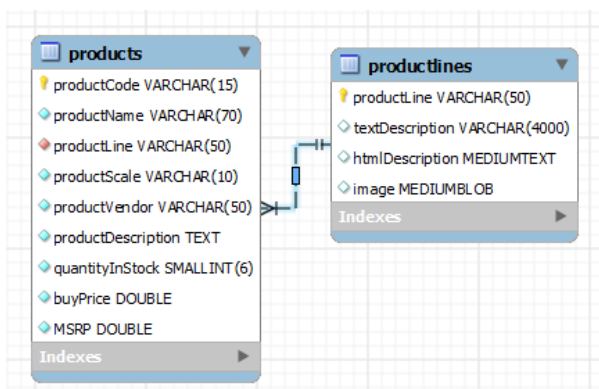
employeeNumber	lastName	firstName	extension	email	officeCode	reportsTo	jobTitle
1102	Bondur	Gerard	x5408	gbondur@classicmodelcars.com	4	1056	Sale Manager (EMEA)
1188	Firelli	Julie	x2173	jfirelli@classicmodelcars.com	2	1143	Sales Rep
1216	Patterson	Steve	x4334	spatterson@classicmodelcars.com	2	1143	Sales 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
1401	Castillo	Pamela	x2759	pcastillo@classicmodelcars.com	4	1102	Sales Rep
1501	Bott	Larry	x2311	lbott@classicmodelcars.com	7	1102	Sales Rep
1504	Jones	Bary	x102	bjones@classicmodelcars.com	7	1102	Sales Rep
1621	Nishi	Mami	x101	mnishi@classicmodelcars.com	5	1056	Sales Rep

Bảng *offices* không còn dòng có *officeCode* = 1

officeCode	city	phone	addressLine1	addressLine2	state	country	postalCode	territory
2	Boston	+1 215 837 0825	1550 Court Place	Suite 102	MA	USA	02107	NA
3	NYC	+1 212 555 3000	523 East 53rd Street	apt. 5A	NY	USA	10022	NA
4	Paris	+33 14 723 4404	43 Rue Jouffroy D'abbans	NULL	NULL	France	75017	EMEA
5	Tokyo	+81 33 224 5000	4-1 Koicho	NULL	Chiyoda-Ku	Japan	102-8578	Japan
6	Sydney	+61 2 9264 2451	5-11 Wentworth Avenue	Floor #2	NULL	Australia	NSW 2010	APAC
7	London	+44 20 7877 2041	25 Old Broad Street	Level 7	NULL	UK	EC2N 1HN	EMEA
8	Boston	+1 215 837 0825	1550 dummy street	NULL	MA	USA	02107	NA

4. Cập nhật dữ liệu có ràng buộc

Giữa các bảng dữ liệu có thể tồn tại các ràng buộc, ví dụ ràng buộc khóa ngoài giữa bảng *products* và *productlines*.



Nếu chúng ta xóa một dòng dữ liệu trong bảng productline mà vẫn còn tồn tại các dòng dữ liệu trong bảng products tham chiếu tới dòng dữ liệu này, ngầm định sẽ không được phép.

Ví dụ: Xóa các dòng sản phẩm có mã là 'Ships'

```
DELETE FROM productlines  
WHERE productLine='Ships'
```

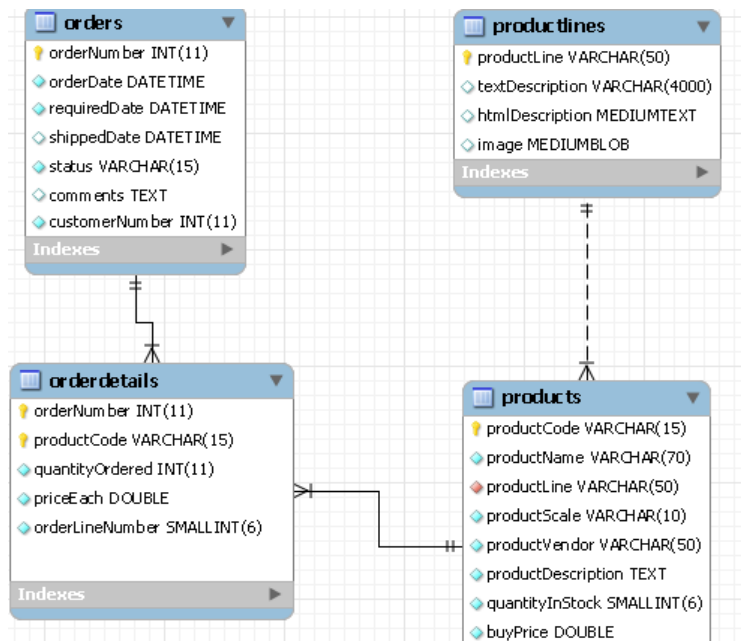
Sẽ hiện thông báo lỗi “Cannot delete or update a parent row: a foreign key constraint fails (classicmodels`.`products`, CONSTRAINT `fk_products_productlines` FOREIGN KEY (productLine`) REFERENCES `productlines` (productLine`) ON DELETE NO ACTION ON UPDATE NO ACTION)”

Nếu khai báo khóa ngoài với tùy chọn ON DELETE CASCADE, hệ thống sẽ tự động xóa các dòng dữ liệu trong bảng products tham chiếu tới dòng dữ liệu này.

Nếu khai báo khóa ngoài với tùy chọn ON DELETE SET NULL, thì khóa ngoài productLine của các dòng tham chiếu sẽ được thiết lập là NULL.

❖ Bài tập thực hành

1. Thực hành các lệnh INSERT, UPDATE và DELETE trên các bảng trong hình dưới đây của CSDL *classicmodels*.



2. Tạo một bảng đặt tên là *temp_orderdetails*, sau đó thực hiện thêm dữ liệu trong ngày gần đây nhất từ bảng *orderdetails* vào bảng trên.
3. Sửa các nhân viên có titleJob là 'Sales Rep' thành 'Sales Representative'

Bài thực hành số 10

Mô hình hóa CSDL sử dụng công cụ MySQL Workbench

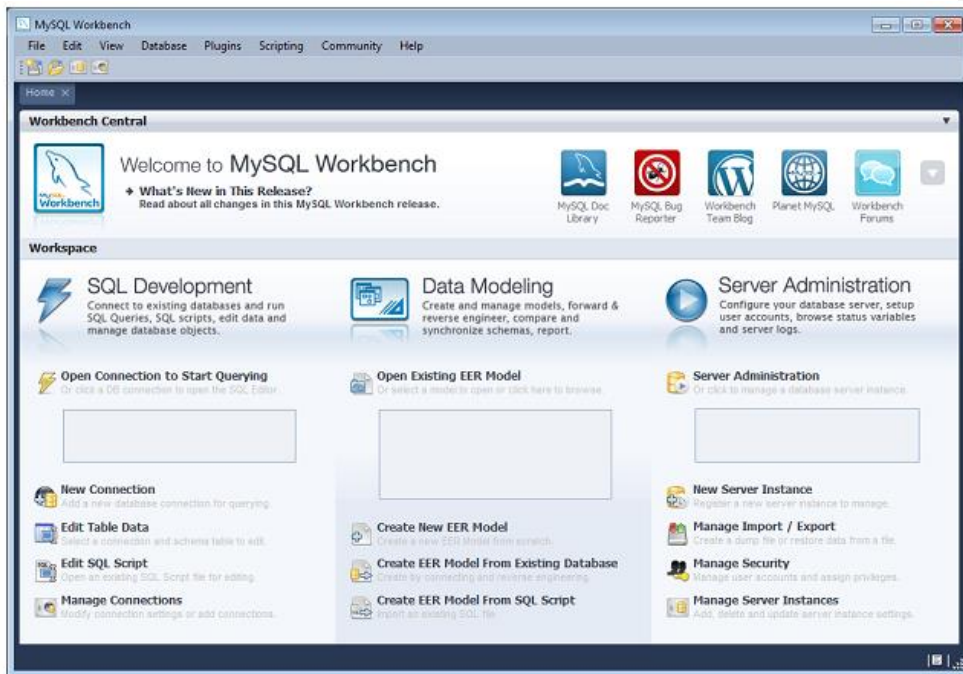
❖ Nội dung chính:

- Giới thiệu MySQL Workbench
- Tạo mô hình EER
- Tạo CSDL từ mô hình quan hệ thực thể EER và ngược lại

1. Giới thiệu MySQL Workbench

MySQL Workbench cung cấp một công cụ đồ họa để làm việc với MySQL Server và CSDL. MySQL Workbench cung cấp ba lĩnh vực chức năng chính:

- **Phát triển SQL:** giúp tạo và quản lý các kết nối tới các CSDL server, cũng như cấu hình các tham số kết nối. MySQL Workbench cũng cung cấp khả năng thi hành các truy vấn SQL trên các kết nối CSDL.
- **Mô hình hóa dữ liệu:** Cho phép tạo các mô hình lược đồ CSDL một cách trực quan. Cung cấp khả năng tạo lược đồ từ một CSDL có sẵn (reverse) hoặc tạo CSDL từ lược đồ (forward). Chức năng Table Editor giúp dễ dàng sửa đổi các bảng, cột, chỉ mục, phân mảnh..
- **Quản trị Server:** Giúp tạo và quản trị các MySQL server.



MySQL Workbench được cung cấp trên các môi trường khác nhau:

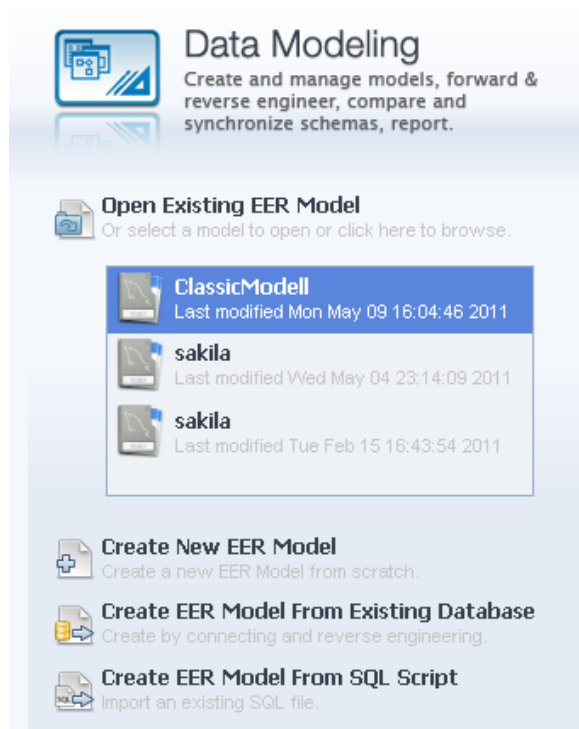
- Windows
- Linux
- Mac OS X

Trên môi trường Windows, để chạy *Workbench* máy tính cần cài đặt .NET framework

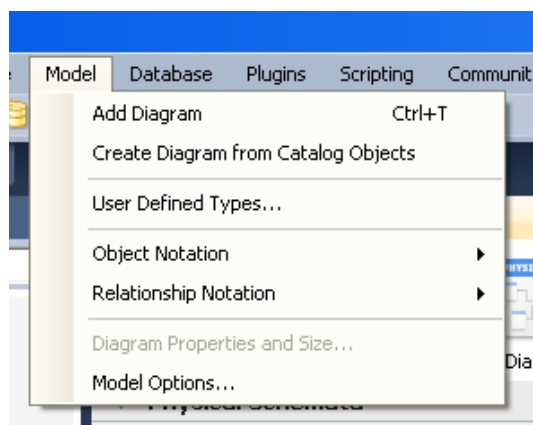
Phần sau sẽ tập trung vào chức năng mô hình hóa dữ liệu

2. Tạo mô hình quan hệ thực thể EER

Bước 1: Sử dụng chức năng *Create new EER Model* để tạo một mô hình mới



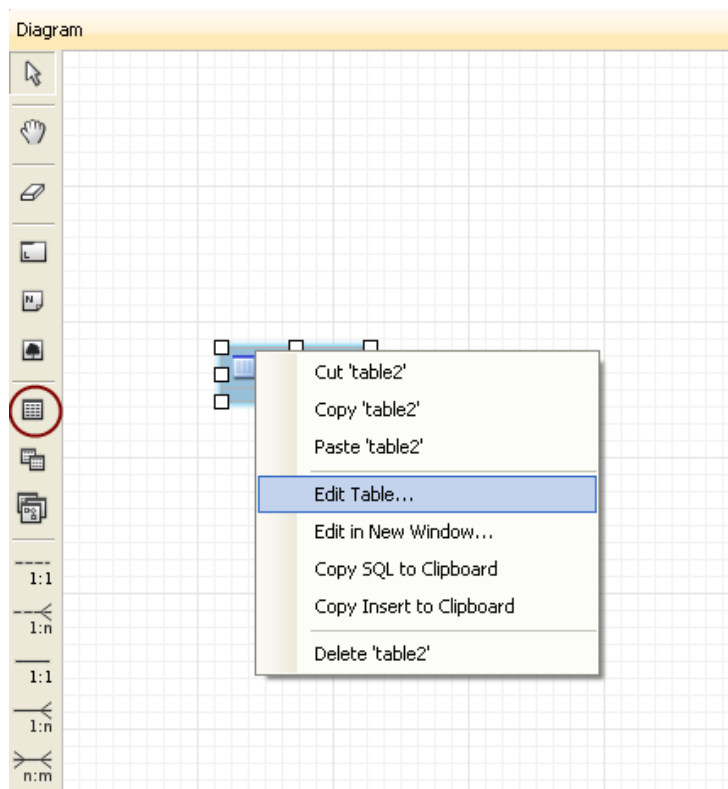
Bước 2: Thêm một biểu đồ mới vào mô hình (chọn *Model* -> *Add Diagram*)



Bước 3: Thêm các bảng yêu cầu vào biểu đồ mới tạo ở bước trước và sửa đổi các bảng để đạt được các yêu cầu đã đặt ra.

Để thêm bảng vào mô hình, chọn vào biểu tượng được khoanh tròn như trong hình dưới.

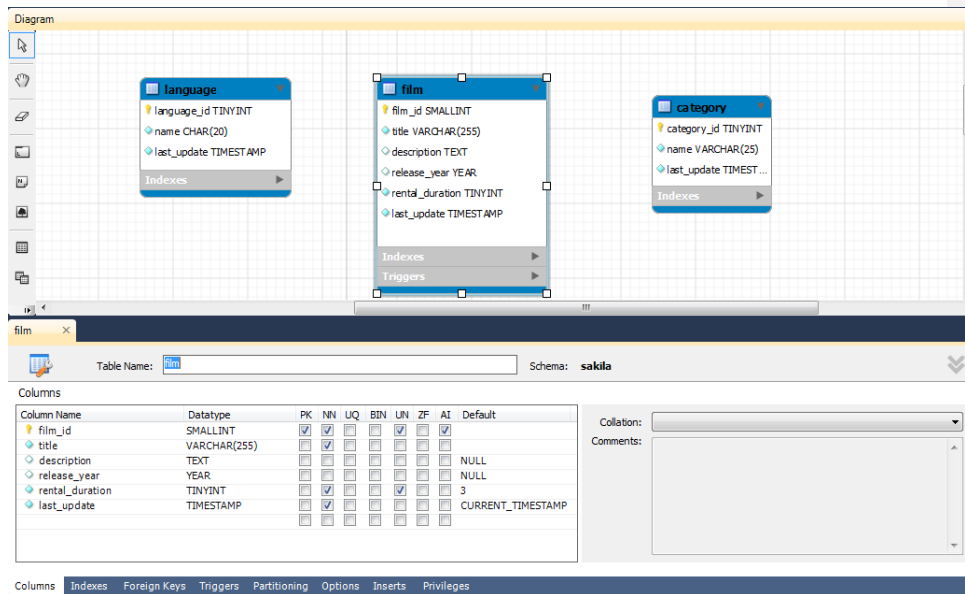
Để sửa đổi bảng, chọn bảng và chọn chức năng Edit Table



Ví dụ: sửa tên bảng mới tạo là *film* và bổ sung thêm các cột như hình vẽ dưới

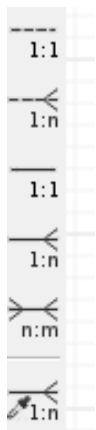
- PK: chỉ thuộc tính là khóa chính
- NN: giá trị không được để trống
- UQ: ràng buộc giá trị là duy nhất
- BIN: để chỉ giá trị lưu ở dạng nhị phân
- UN: Unsigned chỉ thuộc tính lưu ở dạng không dấu

- AI: Nếu giá trị thuộc tính là tự tăng
- Default: Là giá trị ngầm định của cột



Tạo liên kết giữa các bảng

Công cụ hỗ trợ tạo các mối quan hệ giữa các bảng: gồm quan hệ 1-1, quan hệ 1-n, quan hệ n-m

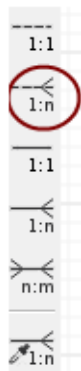


Chú ý: với quan hệ 1-n, công cụ cung cấp 3 tình huống tạo quan hệ:

- *Nếu lựa chọn biểu tượng nét đứt:* một thuộc tính mới sẽ được tự động tạo bên bảng tham chiếu để tham chiếu tới khóa chính của bảng được tham chiếu, và thuộc tính mới tạo ra không phải là thuộc tính khóa chính của bảng tham chiếu.
- *Nếu lựa chọn biểu tượng nét liền:* một thuộc tính mới tương tự như trên được tạo ra, khác biệt ở chỗ thuộc tính này có thuộc tính khóa chính của bảng tham chiếu.
- *Nếu lựa chọn biểu tượng nét liền kèm bút:* sẽ cho phép lựa chọn thuộc tính có sẵn của bảng tham chiếu làm khóa ngoài tham chiếu tới khóa chính của bảng được tham chiếu.

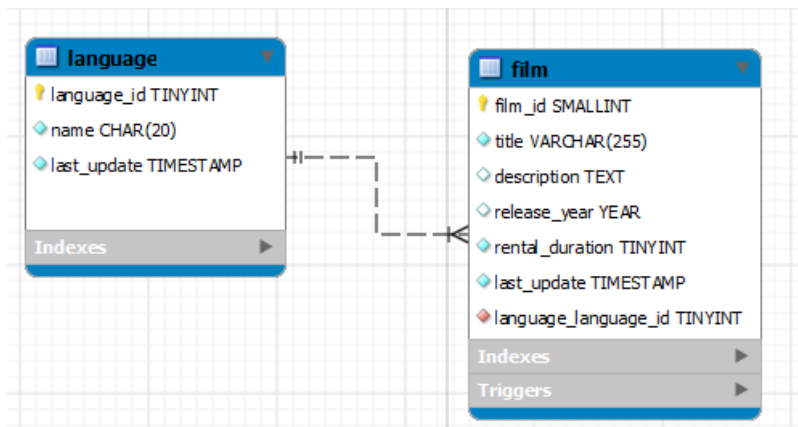
Ví dụ: Tạo quan hệ 1-n giữa bảng language và bảng film đã tạo ở bước trên

Bước 1: Chọn vào biểu tượng như hình vẽ dưới



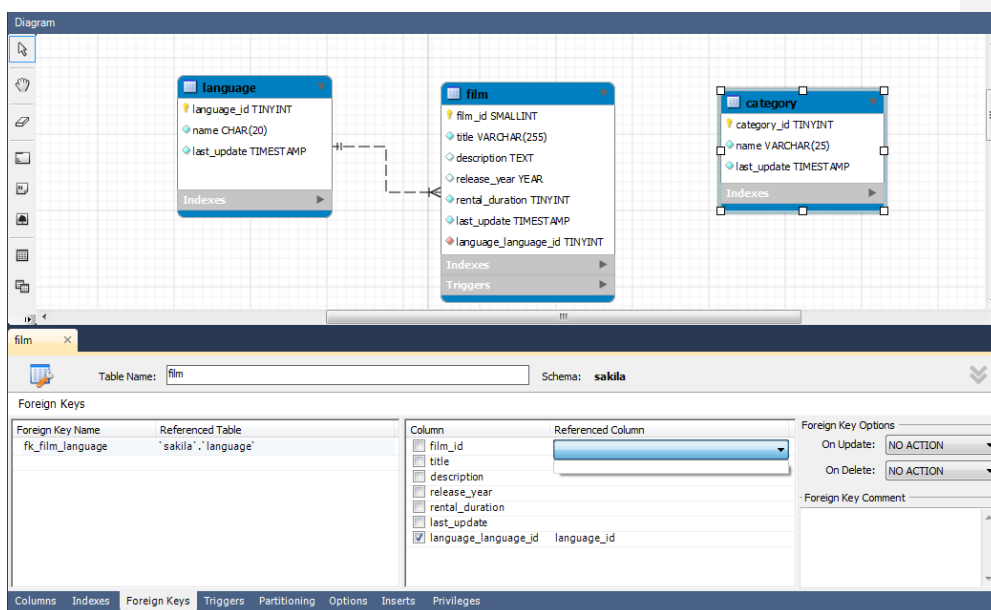
Bước 2: Click chuột vào bảng film, tiếp đó click chuột vào bảng language

Kết quả sẽ sinh ra ràng buộc khóa ngoài liên kết hai bảng film và language. Chú ý: thuộc tính language_language_id sẽ được tự động sinh ra



Ngoài cách tạo liên kết khóa ngoài như trên, có thể tạo liên kết khóa ngoài bằng cách

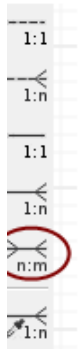
- Chọn sửa đổi bảng tham chiếu
- Chọn vào tab Foreign Keys như hình vẽ dưới đây:



Chú ý: Giao diện này ngoài tạo liên kết khóa ngoài còn hỗ trợ sửa đổi các tùy chọn của khóa ngoài như ON UPDATE, ON DELETE.

Ví dụ: Tạo liên kết n-m giữa hai bảng *film* và *category*

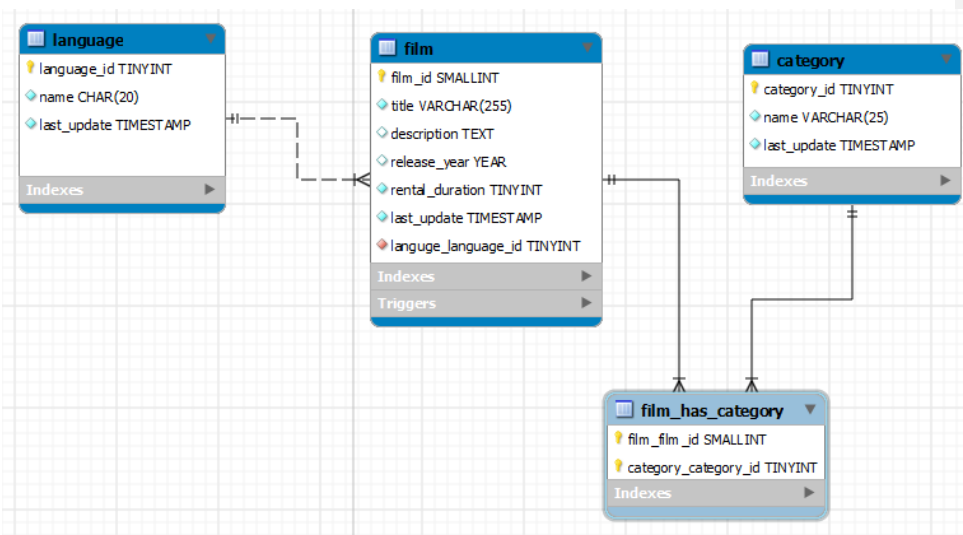
Bước 1: Chọn vào biểu tượng như hình vẽ dưới



Bước 2: Click chuột vào bảng *film* và sau đó là bảng *category*.

Kết quả công cụ sẽ tự động sinh ra một bảng mới có tên *film_has_category* có khóa chính là là tổ hợp từ khóa chính của hai bảng *film* và bảng *category*.

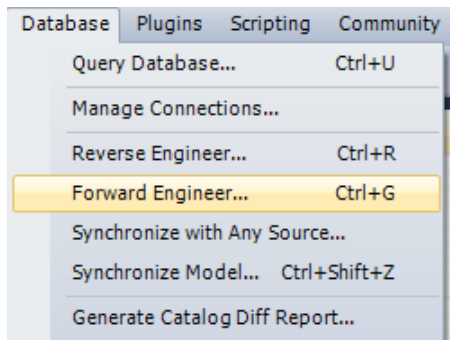
Sau bước tạo trên, người sử dụng có thể sửa đổi bảng mới sinh theo nhu cầu của mình.



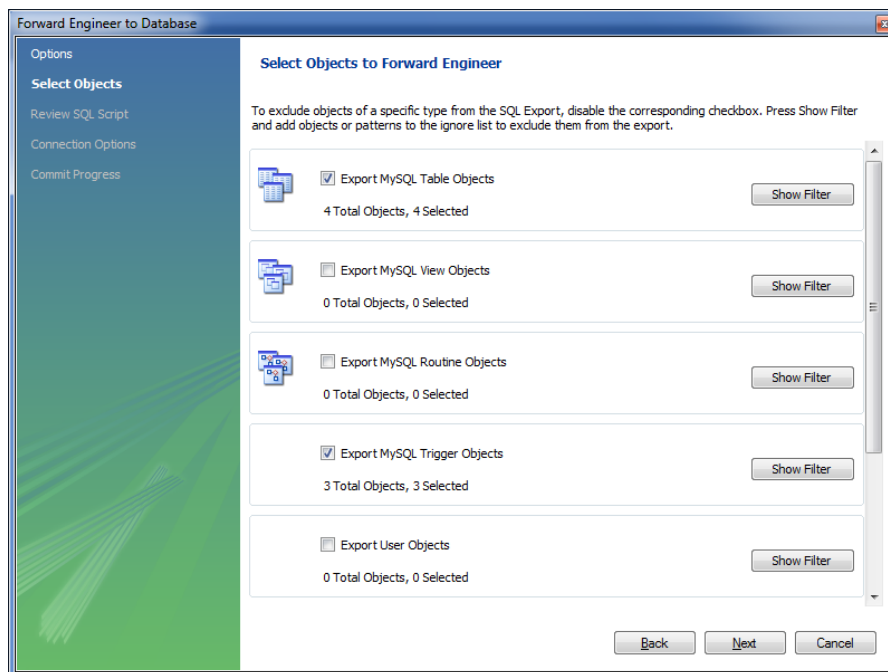
3. Tạo CSDL từ mô hình quan hệ thực thể EER

Để tạo cơ sở dữ liệu mới tên là **my_classicmodels** lưu vào MySQL từ mô hình trên:

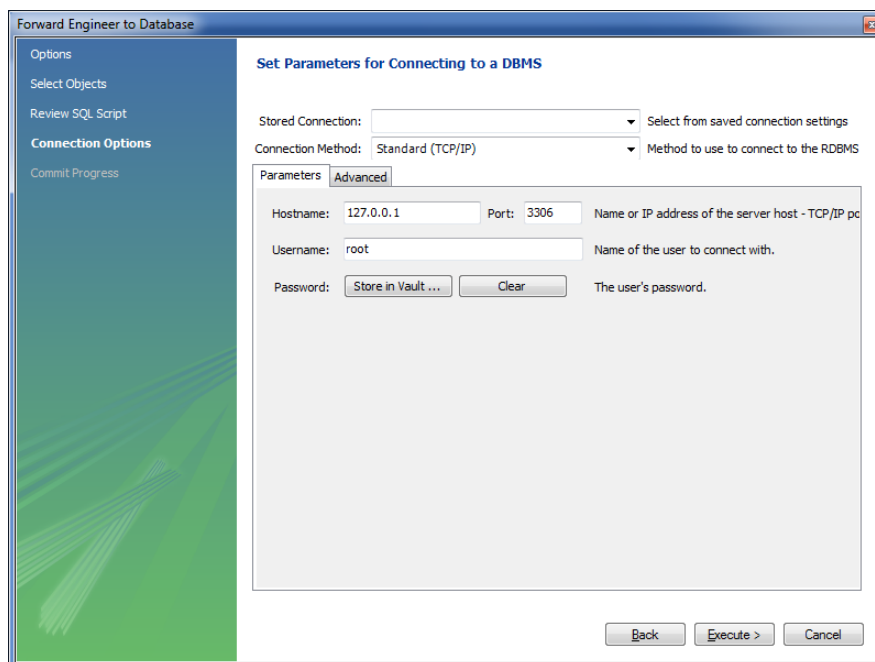
Bước 1: Sử dụng chức năng Database -> Forward Engineer



Bước 2: Chọn các đối tượng từ mô hình EER sẽ lưu vào CSDL

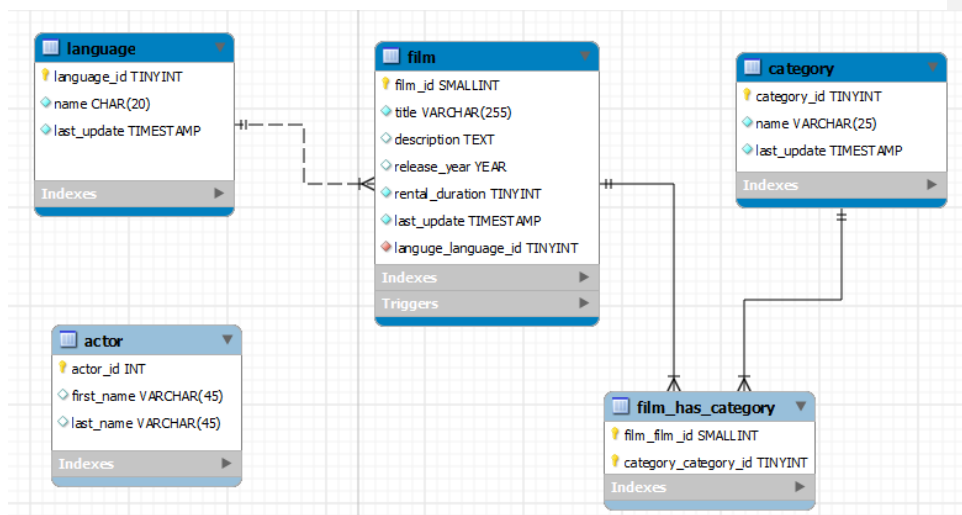


Bước 3: Chọn kết nối tới MySQL server dùng để lưu trữ CSDL sẽ được tạo ra



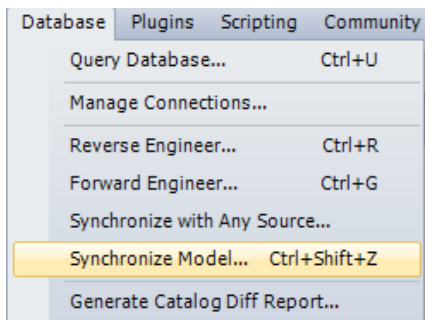
4. Đồng bộ hóa mô hình EER với CSDL trong MySQL Server

Trong quá trình phát triển, mô hình EER hoặc CSDL có sự thay đổi, Workbench cung cấp chức năng hỗ trợ đồng bộ hóa các thay đổi giữa mô hình EER và CSDL.

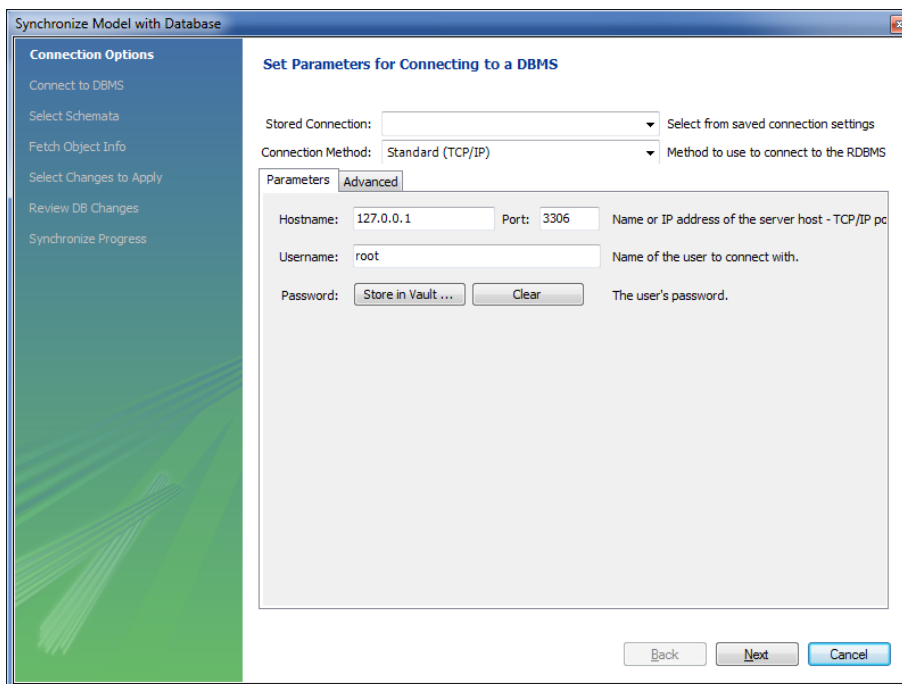


Ví dụ trên, mô hình EER được bổ sung bảng *actor*. Để tiến hành đồng bộ hóa, thực hiện các bước sau:

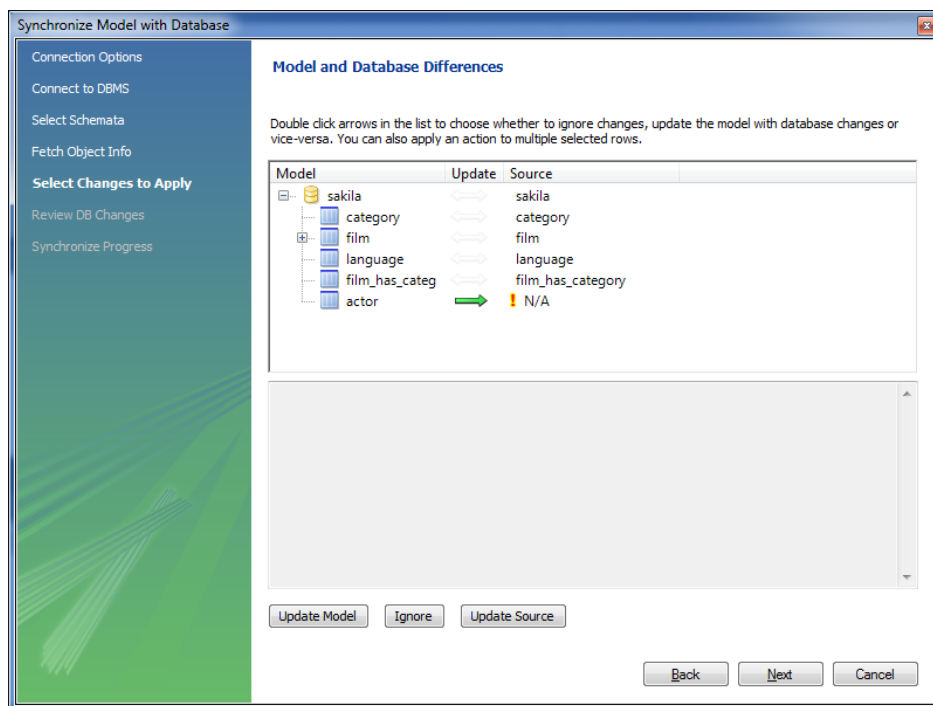
Bước 1: Chọn chức năng Database -> Synchronize Model



Bước 2: Chọn kết nối tới MySQL server cần đồng bộ hóa

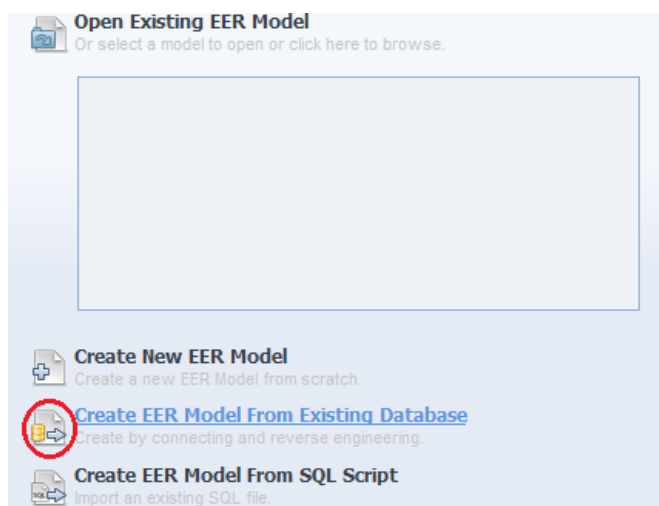


Bước 3: Chọn CSDL muốn đồng bộ và đối tượng cần đồng bộ hóa giữa mô hình EER và CSDL

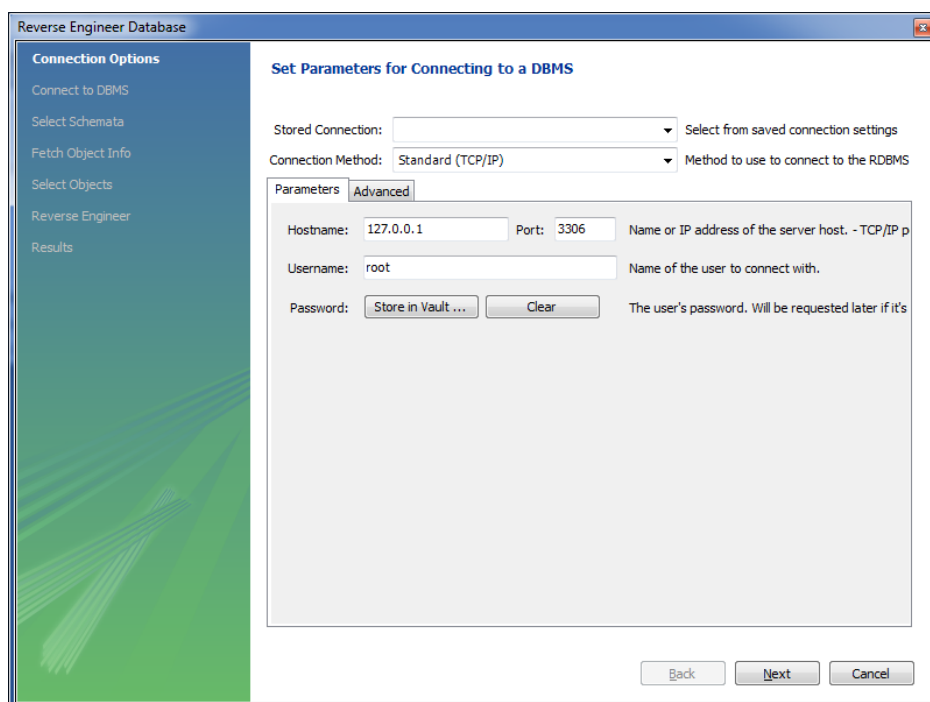


5. Tạo mô hình quan hệ thực thể EER từ CSDL có sẵn

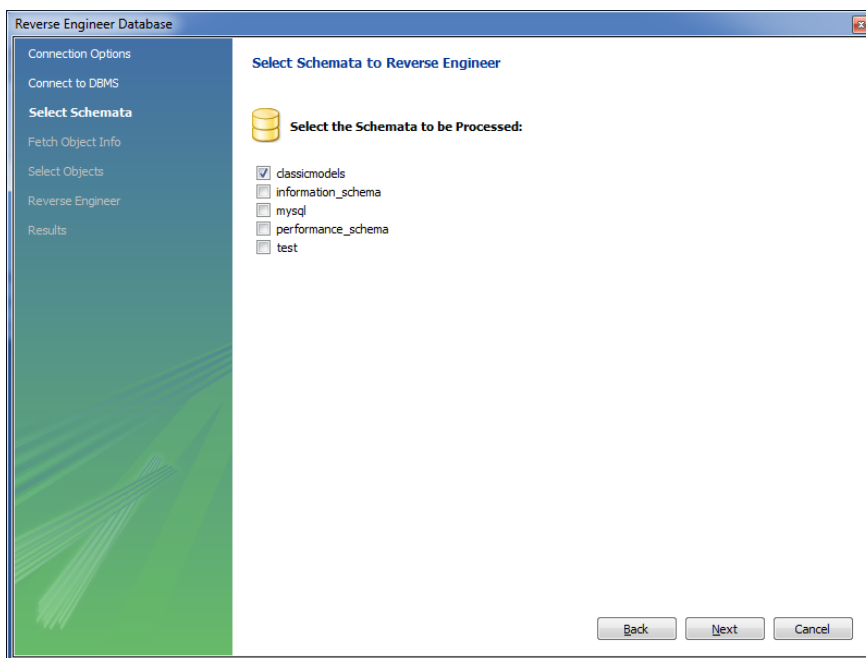
Bên cạnh tạo mô hình quan hệ thực thể EER từ đầu, có thể tạo mô hình từ một CSDL có sẵn, điều này có thể gặp khi cần phát triển tiếp trên một hệ thống CSDL đã có sẵn. Chọn chức năng: “Create EER Model From Existing Database”



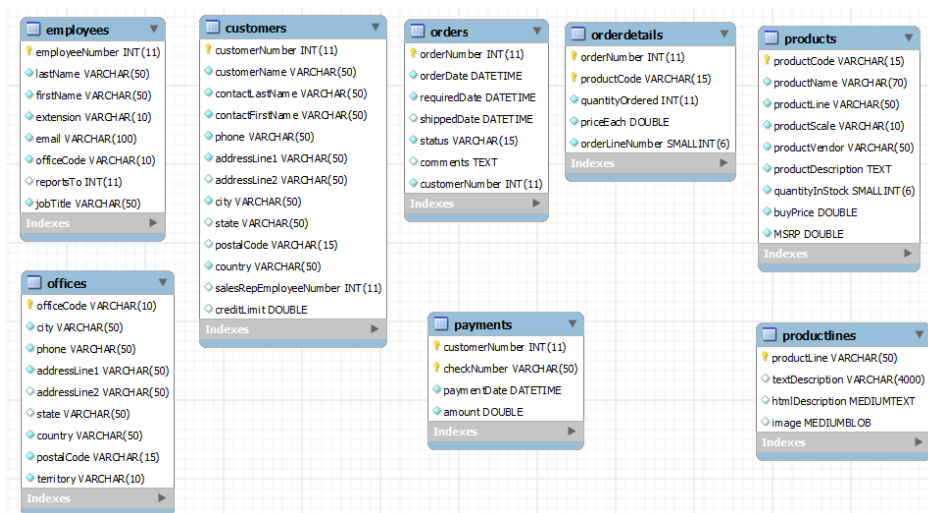
Hộp thoại tiếp theo sẽ chỉ ra Database server muốn kết nối đến



Bước tiếp theo chọn CSDL muốn sinh mô hình EER

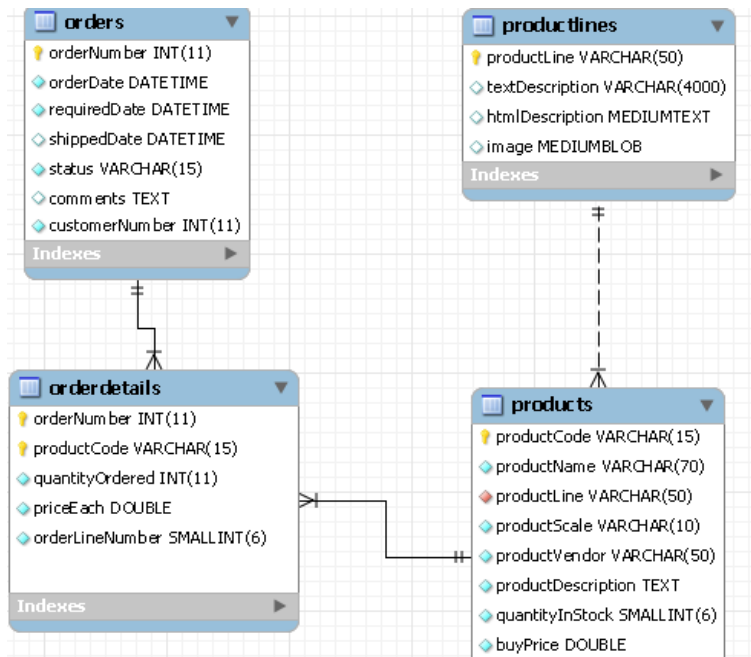


Workbench sẽ tạo ra một mô hình EER từ CSDL được chọn như hình dưới đây. *Lưu ý:* giữa các bảng của mô hình chưa có liên kết do trong CSDL gốc, chưa có liên kết giữa các bảng dữ liệu.



❖ Bài tập thực hành

1. Tạo một mô hình mới tên là `my_classicmodels` gồm các bảng sau:



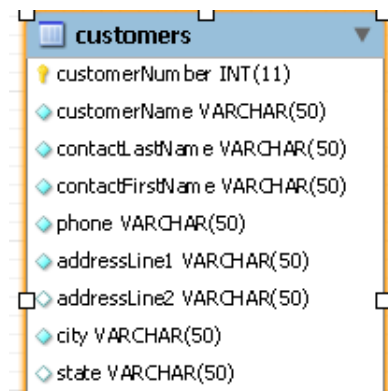
Các ràng buộc khóa ngoài với tùy chọn ON UPDATE CASCADE

Các bảng sử dụng engine InnoDB.

Các khóa chính đều là kiểu số tự động tăng

Dùng chức năng Forward Engine để tạo cơ sở dữ liệu đặt tên là **my_classicmodels**

2. Bổ sung các bảng *customers* sau vào mô hình đã tạo ở câu 1



Bảng *orders* đã tạo ở câu 1 sẽ tham chiếu tới bảng *customers*

Sau đó sử dụng chức năng đồng bộ hóa để đồng bộ mô hình với CSDL **my_classicmodels** lưu trong MySQL Server.