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
```

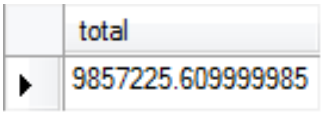
Kết quả trả về như sau:

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:



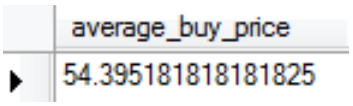
	total
▶	9857225.609999985

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

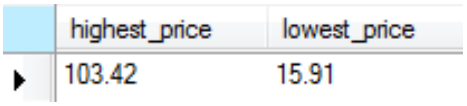
Hàm MAX trả về giá trị lớn nhất và hàm MIN trả về giá trị nhỏ nhất của một tập các giá trị.

```
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:



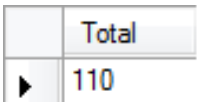
highest_price	lowest_price
103.42	15.91

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 kiệ 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 di 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 itemsCount,
     sum(priceEach * quantityOrdered) AS total
FROM orderdetails
GROUP BY ordernumber
```

ordernumber	itemsCount	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	33383.14000000001
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 itemsCount,
     sum(priceEach * quantityOrdered) AS total
FROM orderdetails
GROUP BY ordernumber
HAVING total > 1000
```

ordernumber	itemsCount	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	33383.14000000001
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 itemsCount,
     sum(priceeach) AS total
FROM orderdetails
GROUP BY ordernumber
HAVING total > 1000 AND itemsCount > 600
```

ordernumber	itemsCount	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

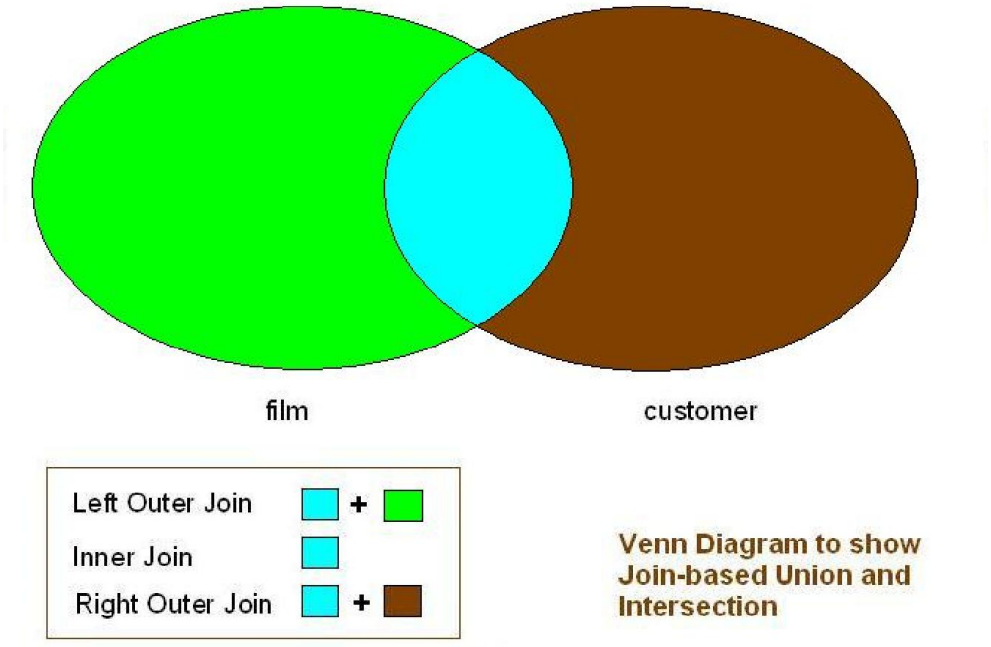
❖ Bài tập thực hành

1. Đưa ra tên các thành phố và số lượng khách hàng tại từng thành phố.
2. Đưa ra số lượng các đơn đặt hàng trong tháng 3/2005.
3. Đưa ra số lượng các đơn đặt hàng trong từng tháng của năm 2005
4. Đưa ra 10 mã đơn đặt hàng có giá trị lớn nhất.
5. 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.

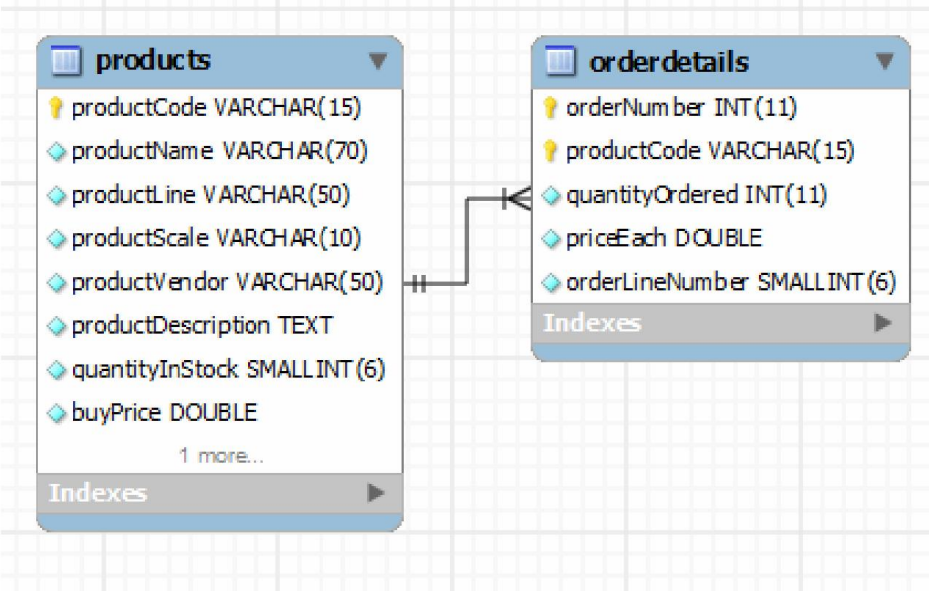
Cú pháp INNER JOIN như sau:

```
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 Souveniers, 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	42015.53999999999
S72_1253	Boeing X-32A JSF	42692.53
S18_4668	1939 Cadillac Limousine	44037.83999999999
S18_2248	1911 Ford Town Car	45306.770000000004
S18_1367	1936 Mercedes-Benz 500K Special Roadster	46078.29
S32_2509	1954 Greyhound Scenicruiser	46519.04999999998
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	137034.21999999994
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+ Petit	86436.96999999999
Auto-Moto Classics Inc.	21554.260000000002
AV Stores, Co.	148410.09000000003
Baane Mini Imports	104224.78999999996
Bavarian Collectables Imports, Co.	31310.089999999997
Blauer See Auto, Co.	75937.76
Boards & Toys Co.	7918.6
CAF Imports	46751.14000000001
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 Corne D'abondance, Co.	33383.14000000001
10115	Classic Legends Inc.	21665.980000000003
10116	Royale Belge	1627.56

2. PHÉP NỐI TRÁI (LEFT JOIN)

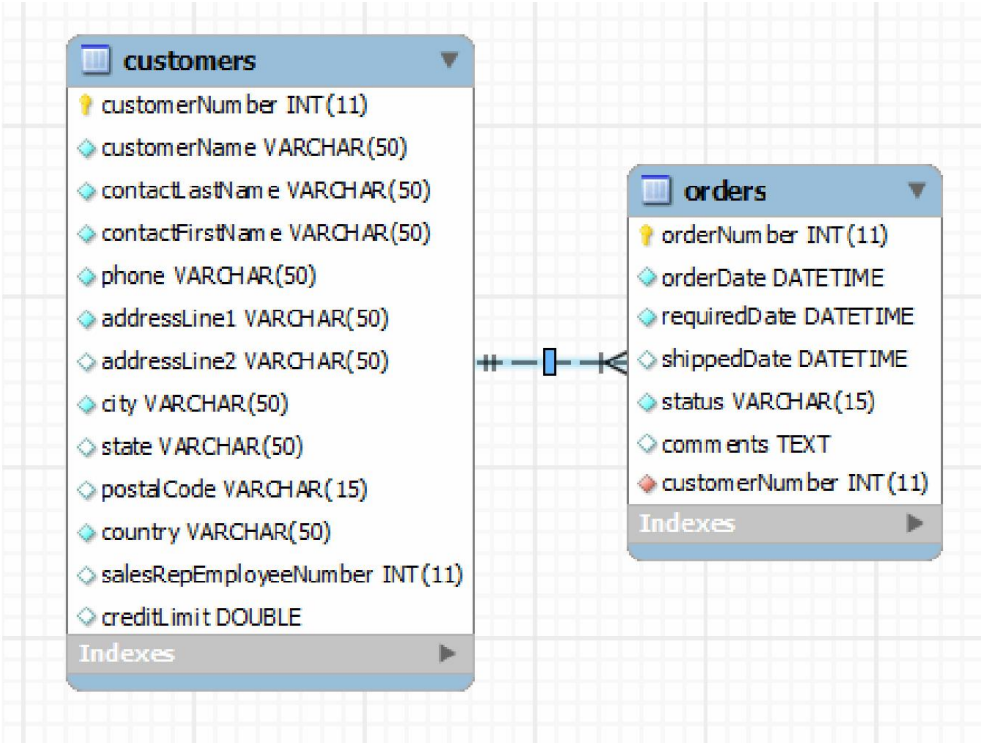
LEFT JOIN cũng là một tùy chọn của câu lệnh SELECT cho phép lấy thêm dữ liệu từ các bảng khác. LEFT JOIN bao gồm các từ khóa LEFT JOIN, tiếp theo là bảng thứ hai muốn thực hiện nối. Yếu tố tiếp theo là từ khóa ON và theo sau bởi các điều kiện nối.

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	customerNumbe
1002	Murphy	Diane	x5800	dmurphy@classicmodelcars.com	1	NULL	NULL
1056	Patterson	Mary	x4611	mpatterso@classicmodelcars.com	1	1002	NULL
1076	Firrelli	Jeff	x9273	jfirrelli@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	Firrelli	Jeff	x9273	jfirrelli@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
Firrelli Jeff	jfirrelli@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
Firrelli Julie	jfirrelli@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...	ghernande@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:

1. Đưa ra thông tin về các nhân viên và tên văn phòng nơi họ làm việc.
2. Đưa ra thông tin về tên khách hàng và tên các sản phẩm họ đã mua.
3. Đưa ra thông tin về các mặt hàng chưa có ai đặt mua.
4. Đưa ra các đơn hàng trong tháng 3/2005 (gồm orderDate, requiredDate, Status) và tổng giá trị của mỗi đơn hàng .
5. 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 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 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 thi hành truy vấn bên ngoài.

Ví dụ: đưa các 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	Tumable 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 LanciaA 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	Tumable 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	Tumable front wheels;

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 thi 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, officia
	S10_4962	1962 LanciaA 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 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...	Turnable 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...	Turnable 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...	Turnable 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 SSK
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 Hornet 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 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 Hornet 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.
6. Với mỗi khách hàng, đưa ra tổng số tiền hàng, và tổng số tiền họ đã thanh toán

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 dung. 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 Kioicho	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 ngậm 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:

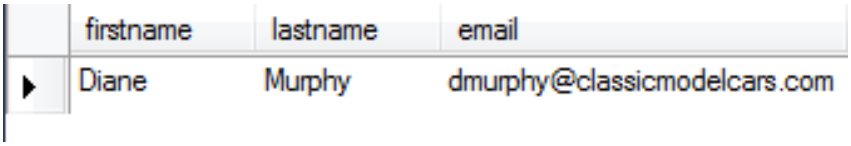
```
UPDATE table_name [, table_name...]
 SETcolumn_name1=expr1
     [, column_name2=expr2 ...]
[WHERE condition]
```

- 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
```

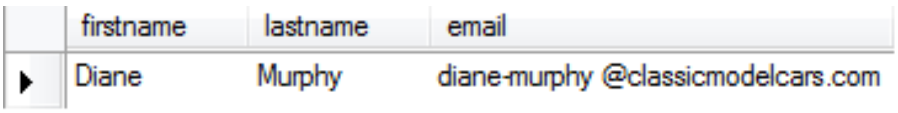


	firstname	lastname	email
▶	Diane	Murphy	dmurphy@classicmodelcars.com

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 DELELE 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	NULL	President
1056	Patterson	Mary	x4611	mpatterso@classicmodelcars.com	1	1002	VP Sales
1076	Firrelli	Jeff	x9273	jfirrelli@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	Firrelli	Julie	x2173	jfirrelli@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*

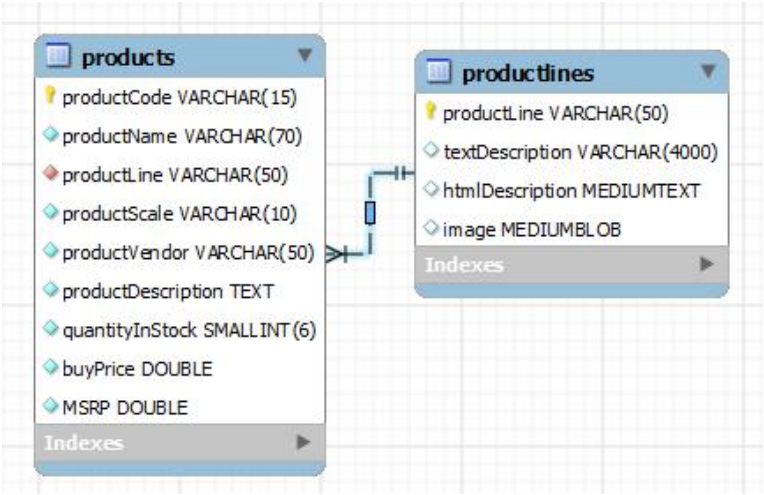
employeeNumber	lastName	firstName	extension	email	officeCode	reportsTo	jobTitle
1102	Bondur	Gerard	x5408	gbondur@classicmodelcars.com	4	1056	Sale Manager (EMEA)
1188	Firrelli	Julie	x2173	jfirrelli@classicmodelcars.com	2	1143	Sales Rep
1216	Patterson	Steve	x4334	spatterson@classicmodelcars.com	2	1143	Sales Rep
1286	Tseng	Foon Yue	x2248	ftseng@classicmodelcars.com	3	1143	Sales Rep
1323	Vanauf	George	x4102	gvanauf@classicmodelcars.com	3	1143	Sales Rep
1337	Bondur	Loui	x6493	lbondur@classicmodelcars.com	4	1102	Sales Rep
1370	Hernandez	Gerard	x2028	ghernande@classicmodelcars.com	4	1102	Sales Rep
1401	Castillo	Pamela	x2759	pcastillo@classicmodelcars.com	4	1102	Sales Rep
1501	Bott	Larry	x2311	lbott@classicmodelcars.com	7	1102	Sales Rep
1504	Jones	Barry	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 Kioicho	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'
```

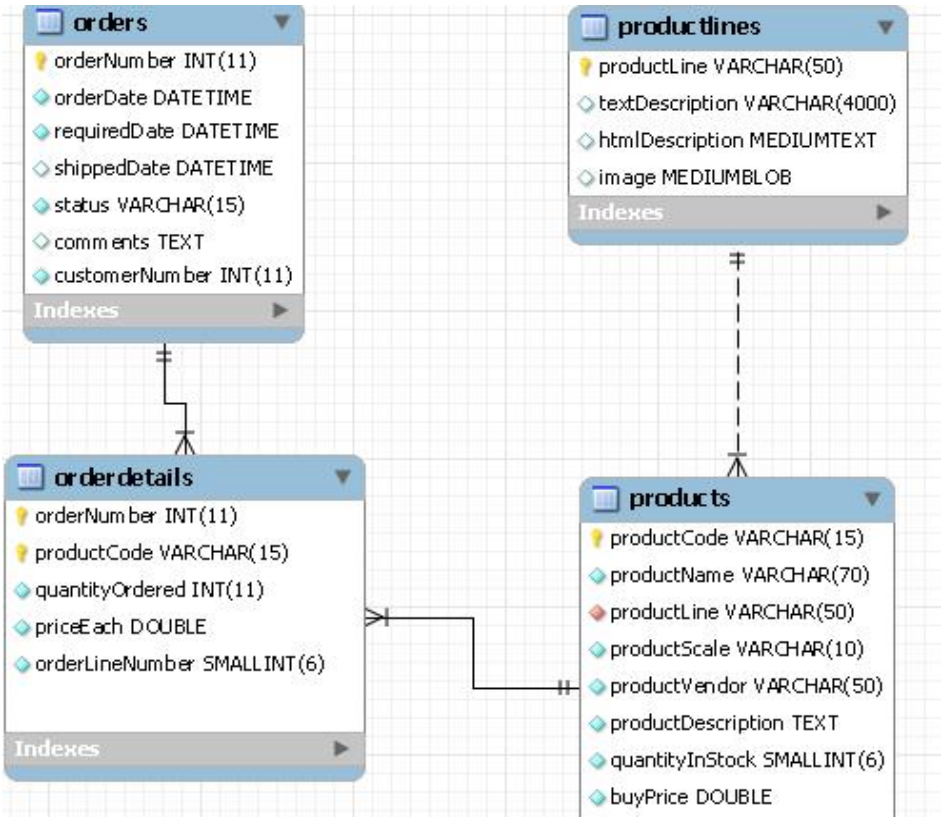
Sẽ hiện thông báo lỗi “Cannot delete or update a parent row: a foreign key constraint fails (`classicmodels`.`products`, CONSTRAINT `fk_products_productlines` FOREIGN KEY (`productLine`) REFERENCES `productlines` (`productLine`) ON DELETE NO ACTION ON UPDATE NO ACTION)”

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

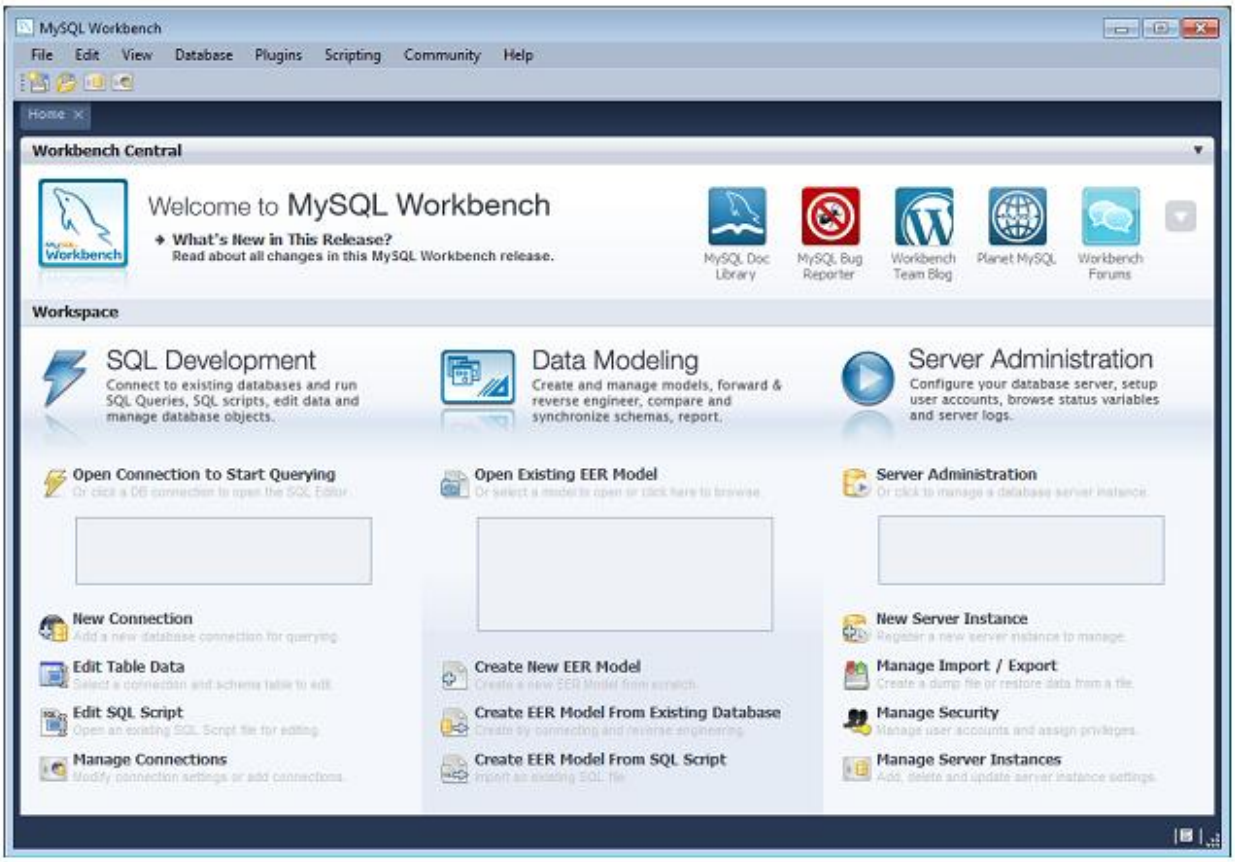
❖ **Nội dung chính:**

- 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

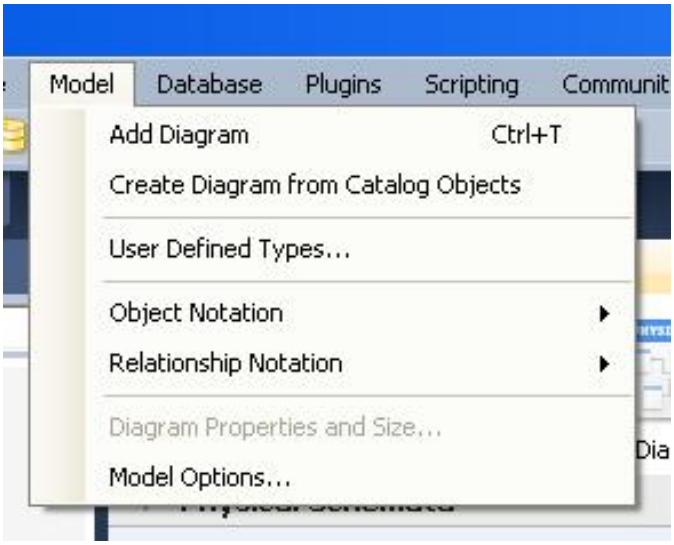
Trên môi trường Windows, để chạy *Workbench* máy tính cần cài đặt .NET framework

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

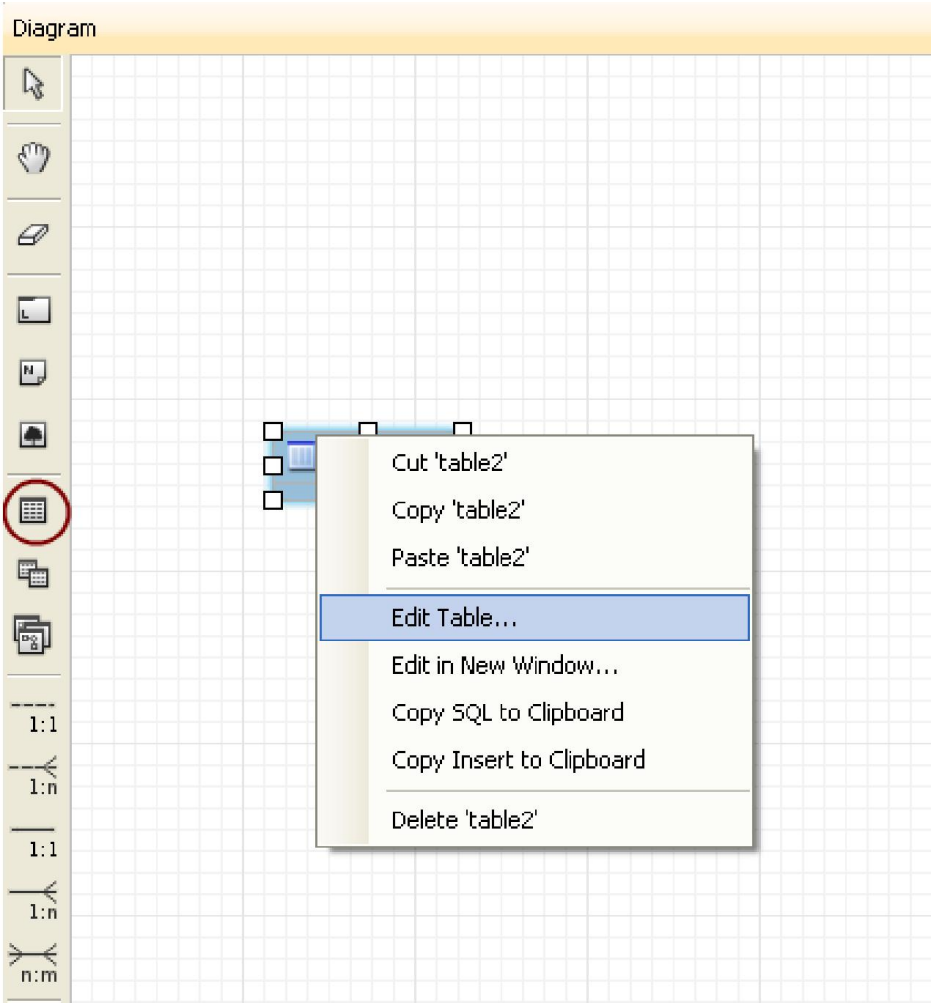
Bước 2: Thêm một biểu đồ mới vào mô hình (chọn *Model -> Add Diagram*)



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.

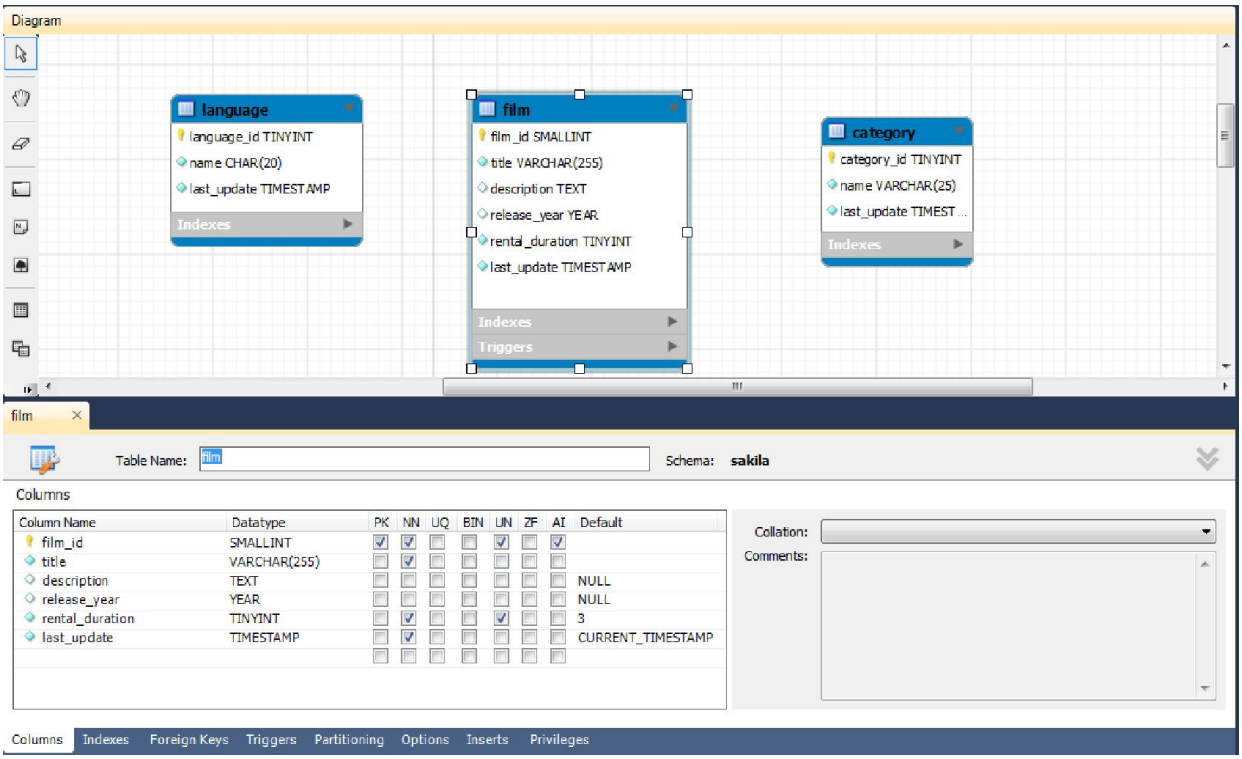
Để sửa đổi bảng, chọn bảng và chọn chức năng Edit Table



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

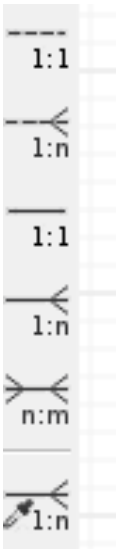
- PK: chỉ thuộc tính là khóa chính
- NN: giá trị không được để trống
- UQ: ràng buộc giá trị là duy nhất
- BIN: để chỉ giá trị lưu ở dạng nhị phân
- UN: Unsigned chỉ thuộc tính lưu ở dạng không dấu

- 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

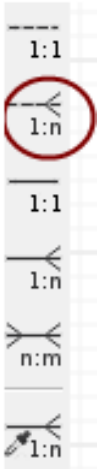


Chú ý: với quan hệ 1-n, công cụ cung cấp 3 tình huống tạo quan hệ:

- *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.

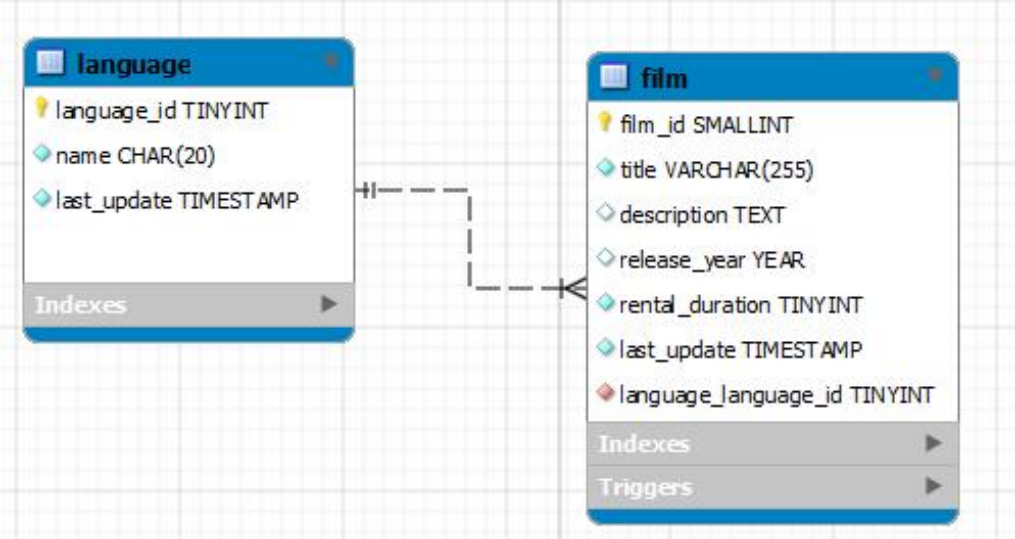
Ví dụ: Tạo quan hệ 1-n giữa bảng language và bảng film đã tạo ở bước trên

Bước 1: Chọn vào biểu tượng như hình vẽ dưới



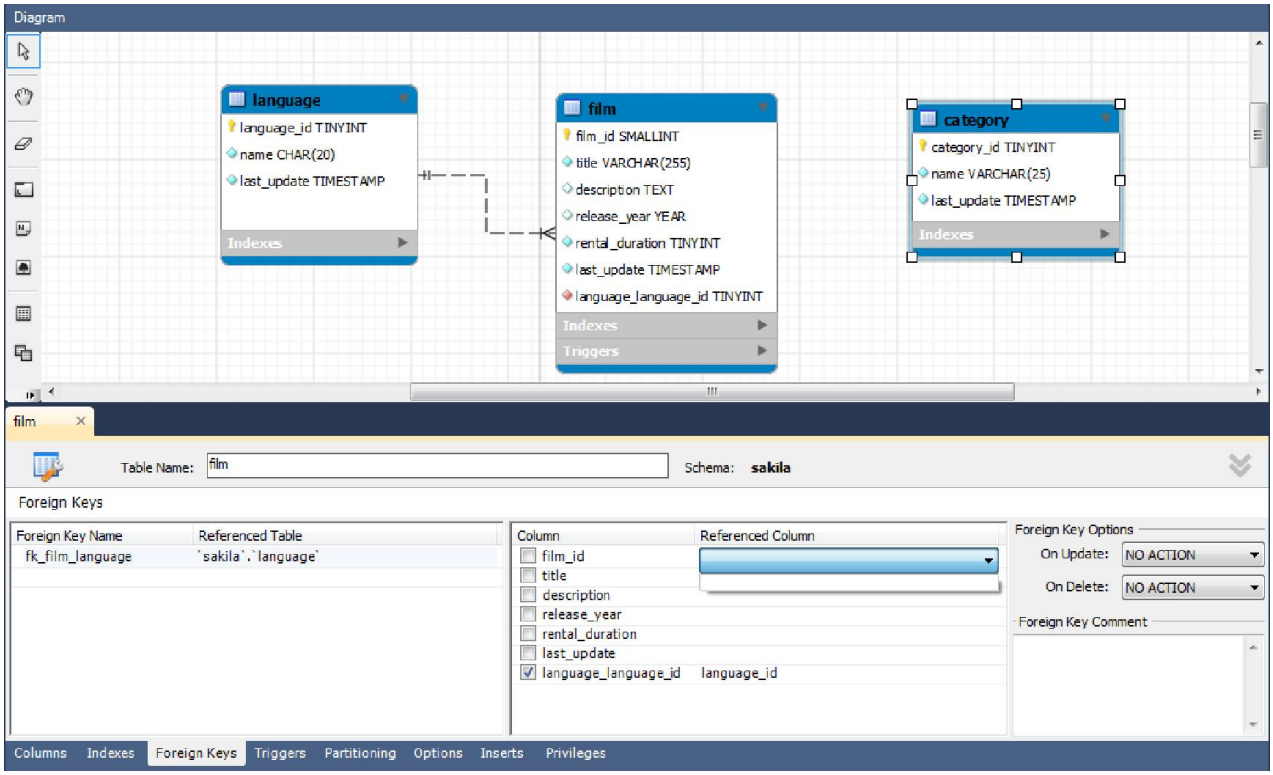
Bước 2: Click chuột vào bảng film, tiếp đó click chuột vào bảng language

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

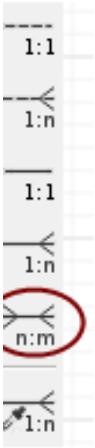
- Chọn sửa đổi bảng tham chiếu
- Chọn vào tab Foreign Keys như hình vẽ dưới đây:



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*

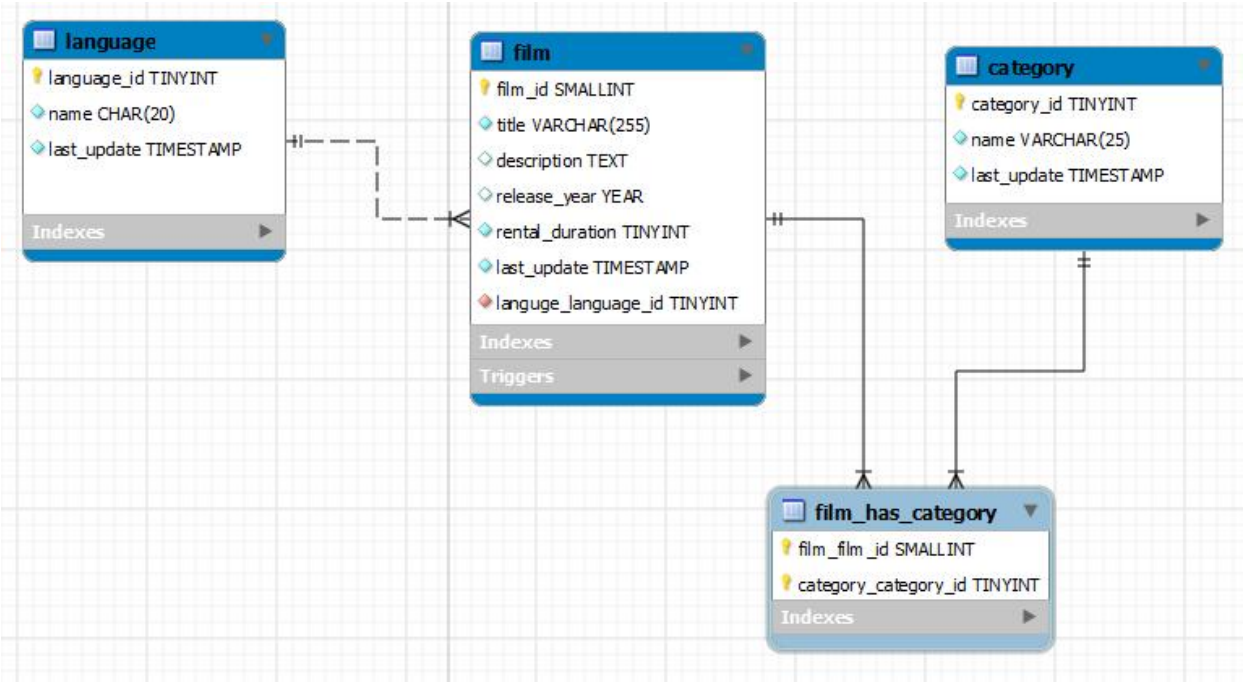
Bước 1: Chọn vào biểu tượng như hình vẽ dưới



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*.

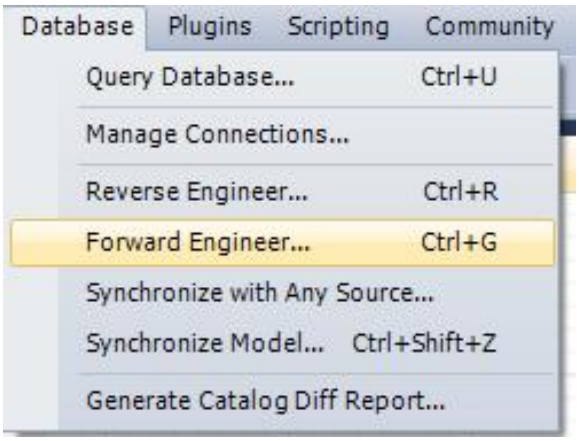
Sau bước tạo trên, người sử dụng có thể sửa đổi bảng mới sinh theo nhu cầu của mình.



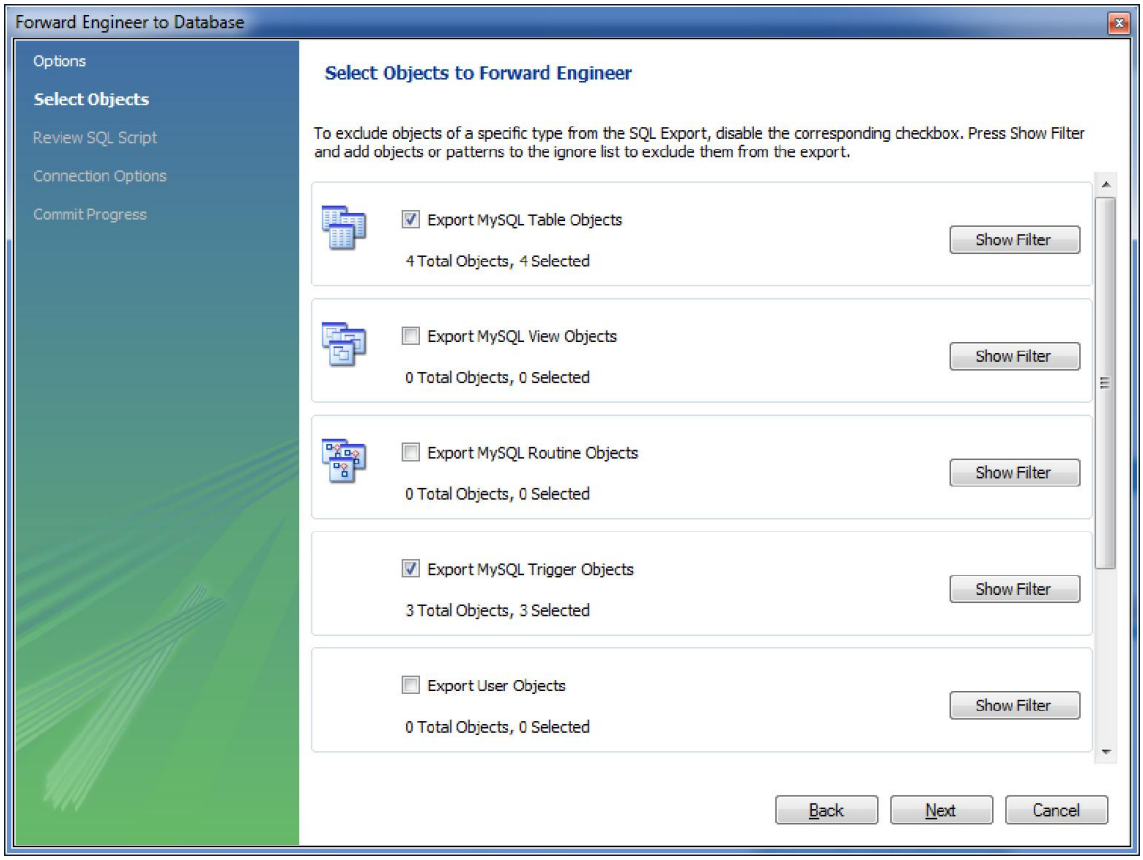
3. Tạo CSDL từ mô hình quan hệ thực thể EER

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

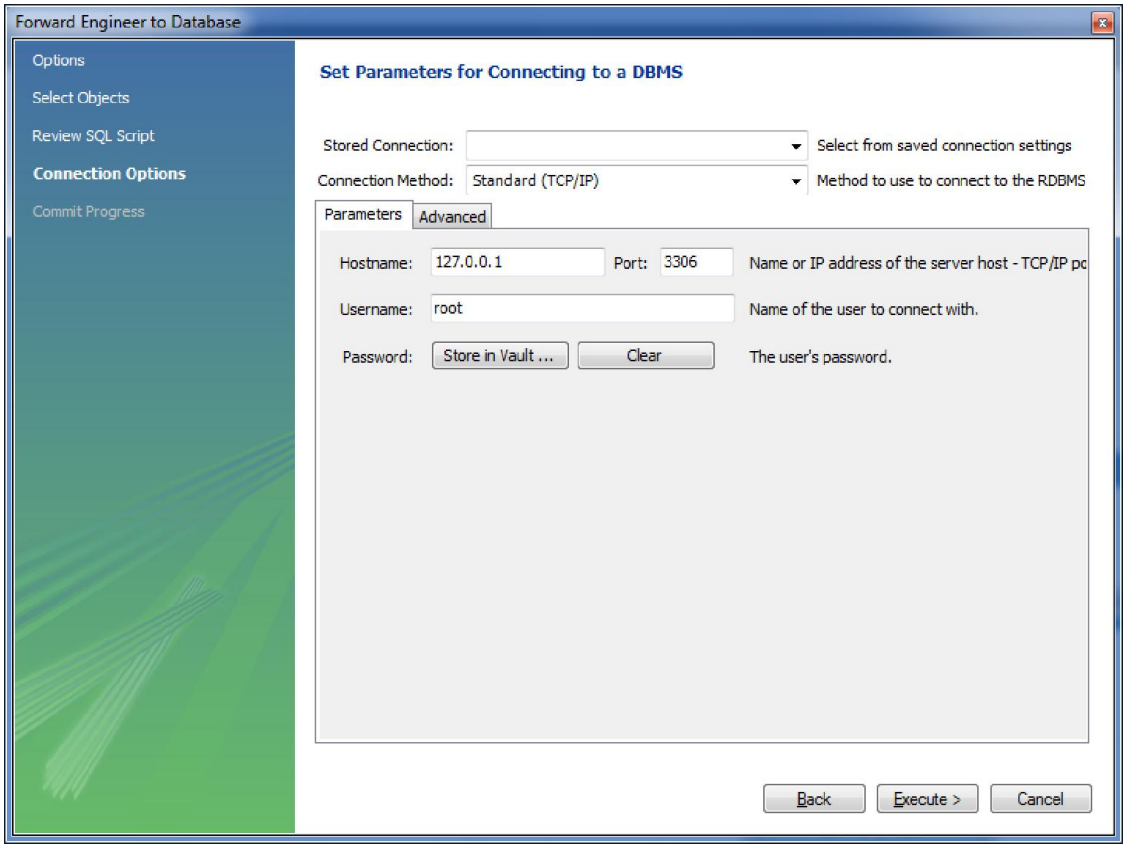
Bước 1: Sử dụng chức năng Database -> Forward Engineer



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

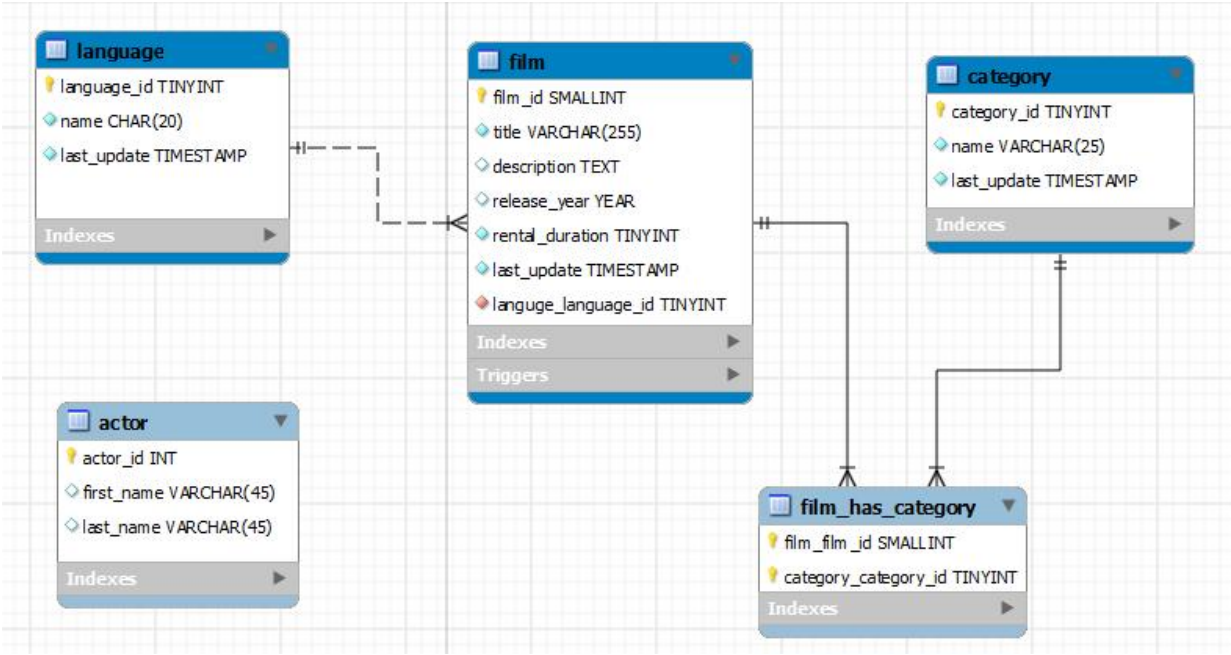


Bước 3: Chọn kết nối tới MySQL server dùng để lưu trữ CSDL sẽ được tạo ra



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

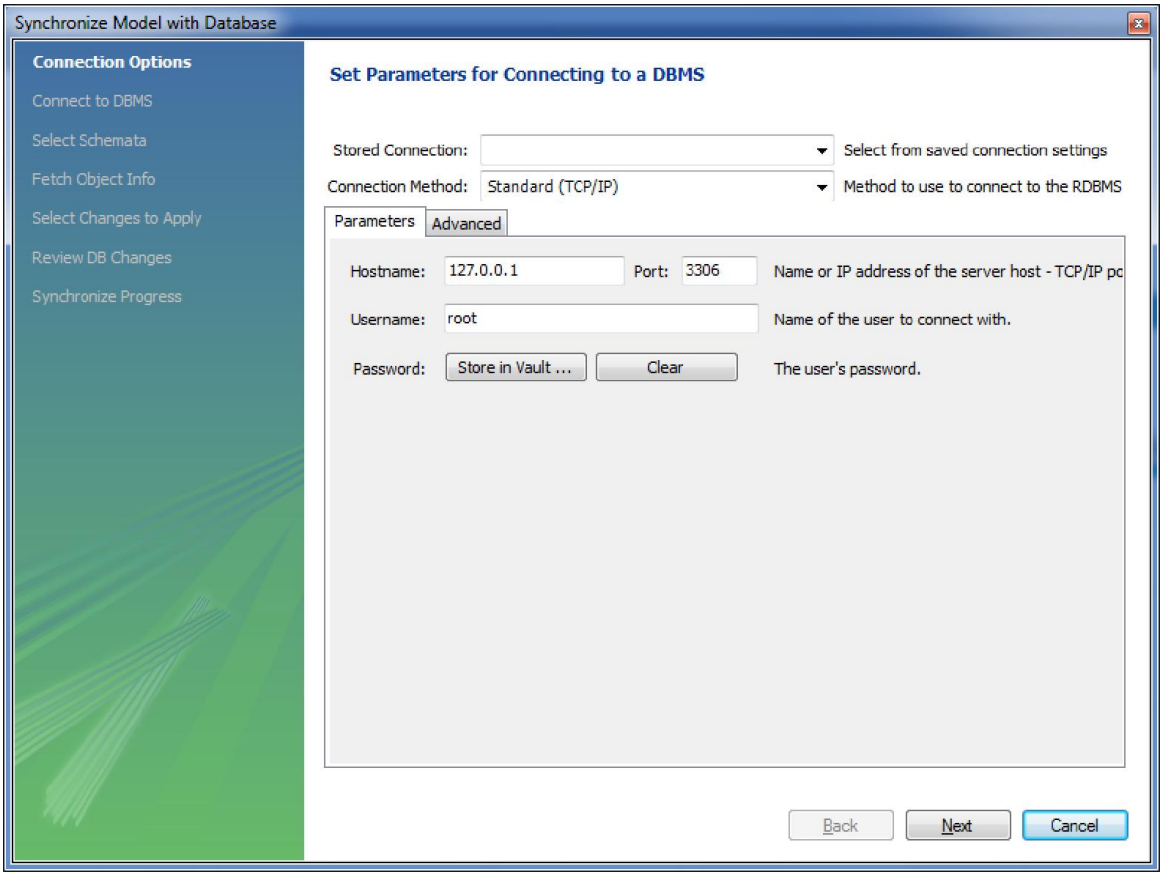
Trong quá trình phát triển, mô hình EER hoặc CSDL có sự thay đổi, Workbench cung cấp chức năng hỗ trợ đồng bộ hóa các thay đổi giữa mô hình EER và CSDL.



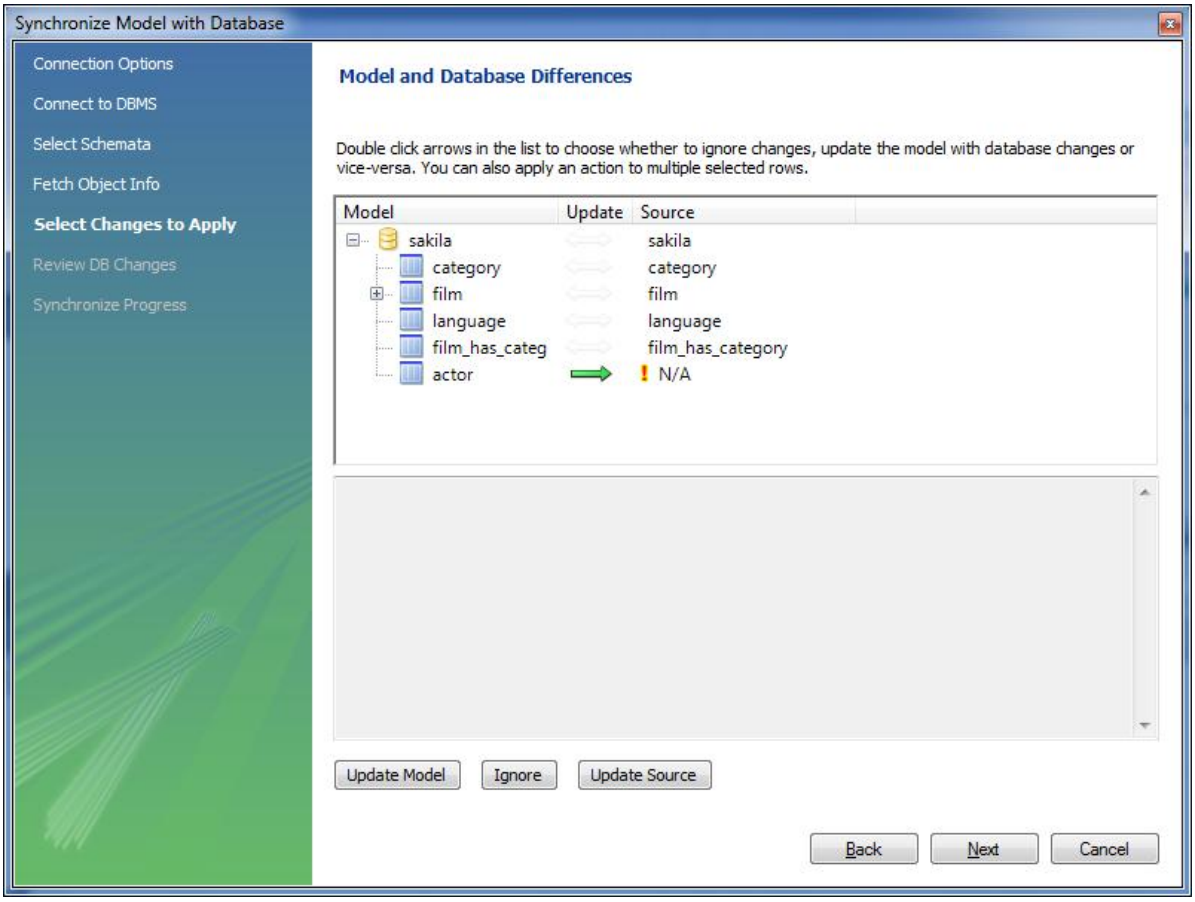
Ví dụ trên, mô hình EER được bổ sung bảng *actor*. Để tiến hành đồng bộ hóa, thực hiện các bước sau:

Bước 1: Chọn chức năng Database -> Synchronize Model

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

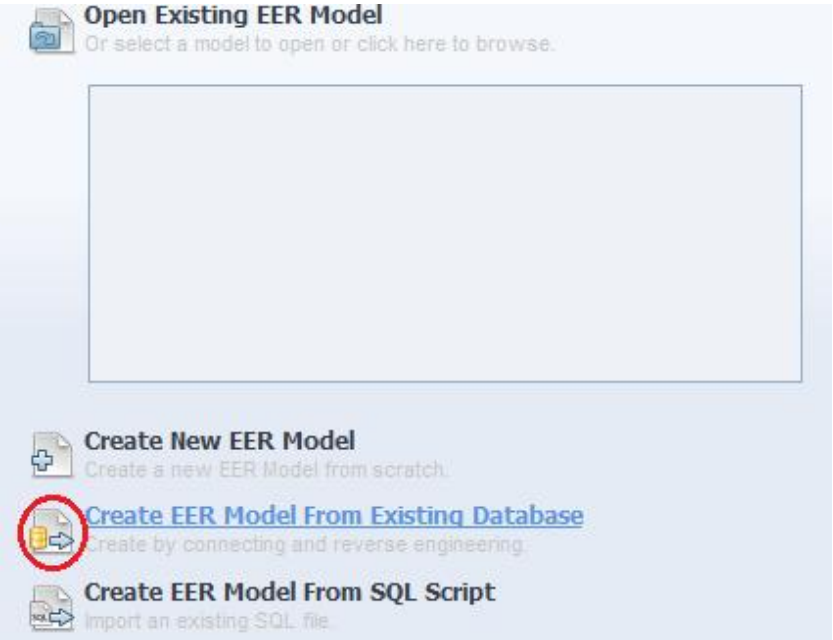


Bước 3: Chọn CSDL muốn đồng bộ và đối tượng cần đồng bộ hóa giữa mô hình EER và CSDL

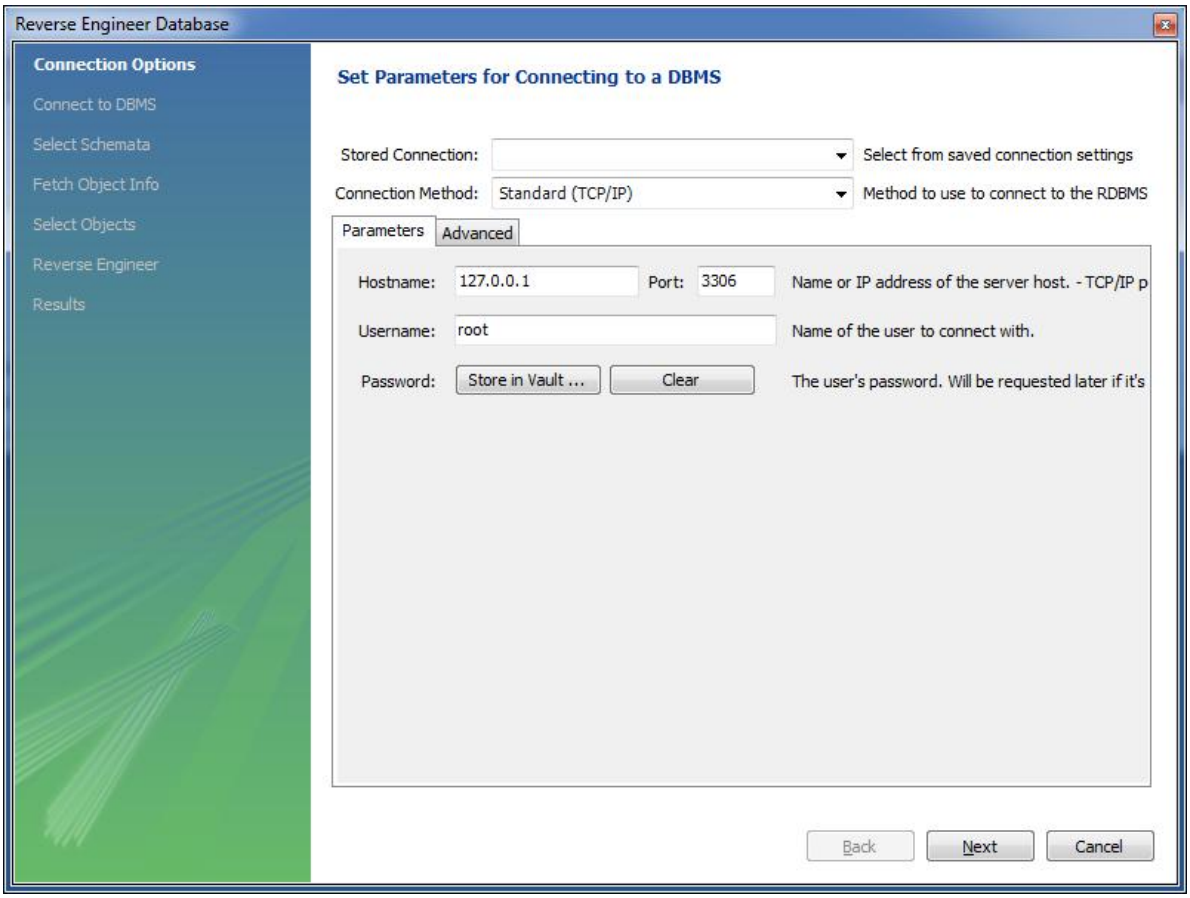


5. Tạo mô hình quan hệ thực thể EER từ CSDL có sẵn

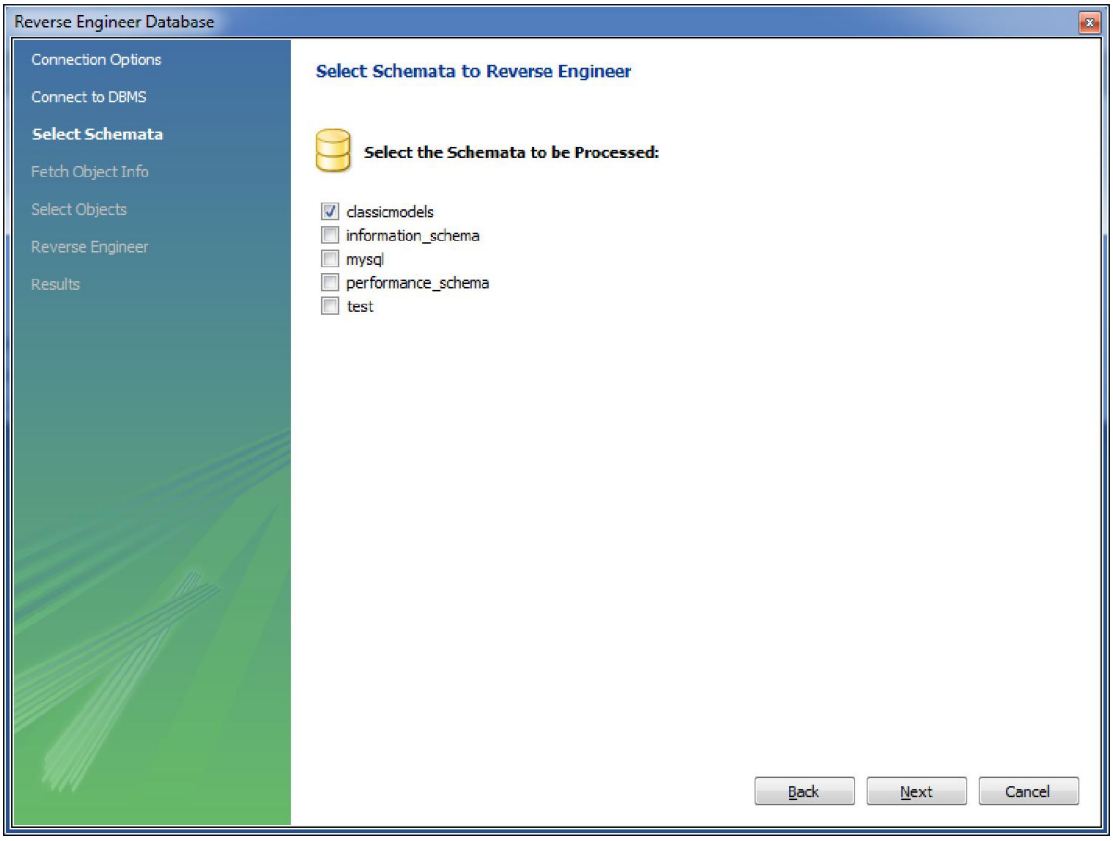
Bên cạnh tạo mô hình quan hệ thực thể EER từ đầu, có thể tạo mô hình từ một CSDL có sẵn, điều này có thể gặp khi cần phát triển tiếp trên một hệ thống CSDL đã có sẵn. Chọn chức năng: “Create EER Model From Existing Database”



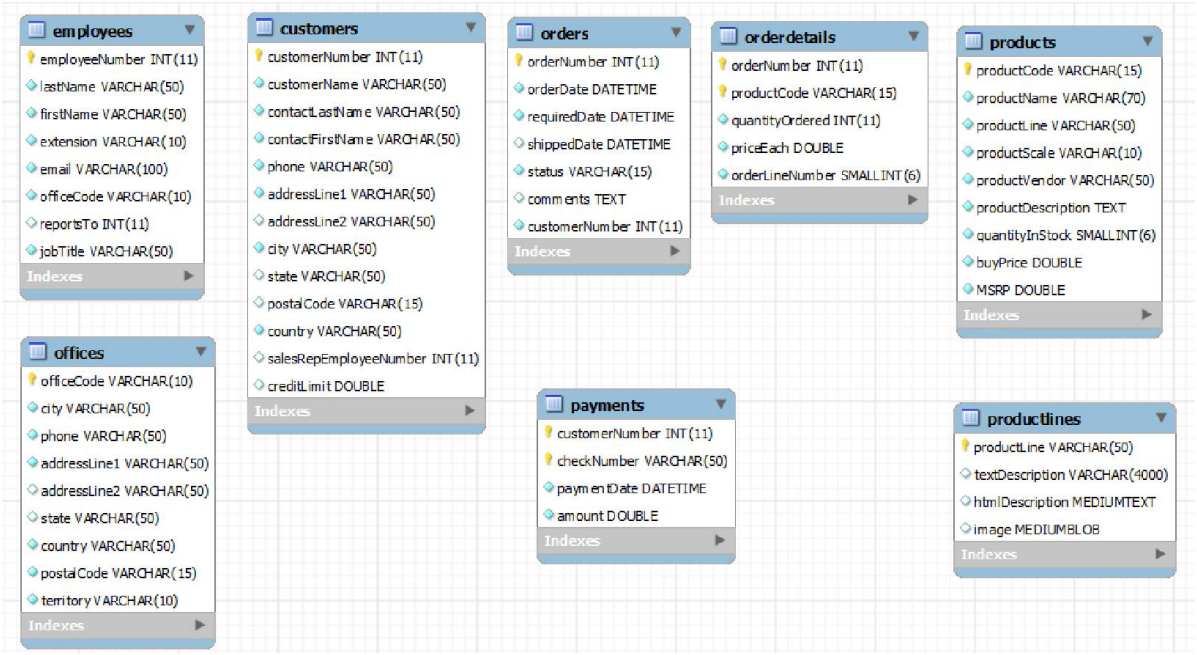
Hộp thoại tiếp theo sẽ chỉ ra Database server muốn kết nối đến



Bước tiếp theo chọn CSDL muốn sinh mô hình EER

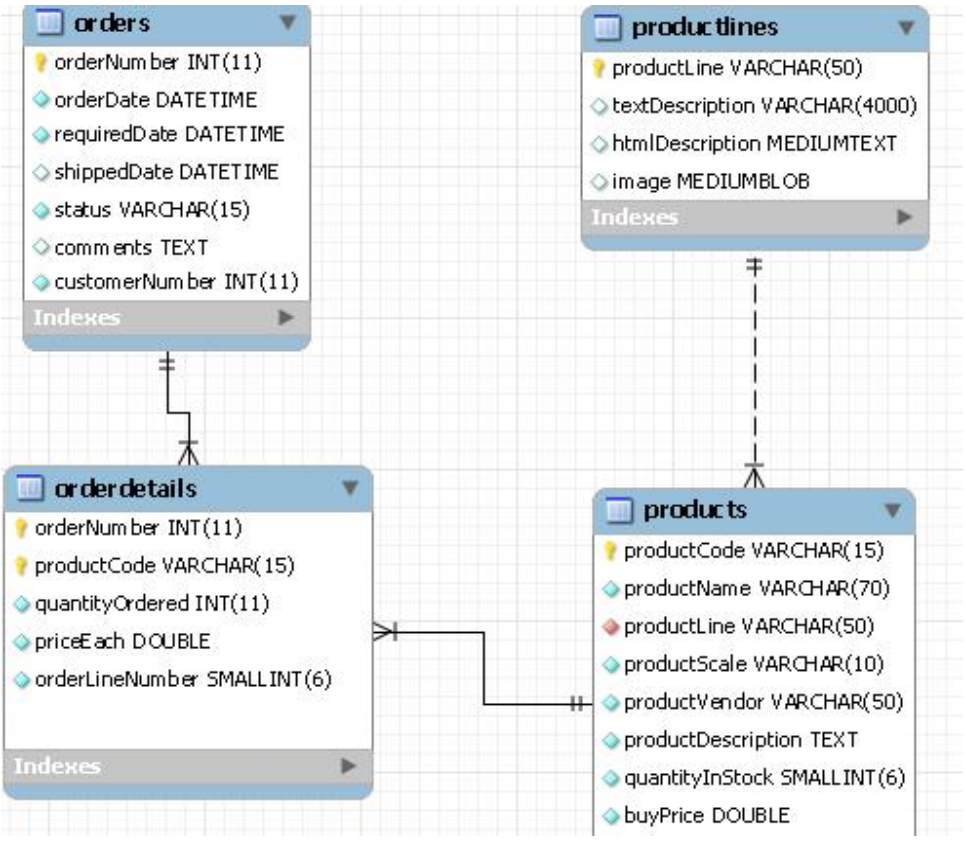


Workbench sẽ tạo ra một mô hình EER từ CSDL được chọn như hình dưới đây. *Lưu ý*: giữa các bảng của mô hình chưa có liên kết do trong CSDL gốc, chưa có liên kết giữa các bảng dữ liệu.



❖ Bài tập thực hành

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



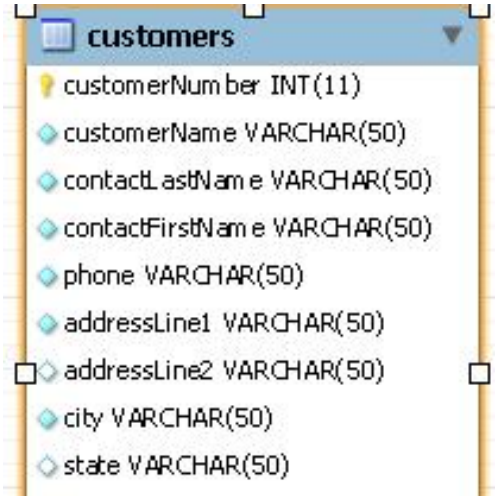
Các ràng buộc khóa ngoài với tùy chọn ON UPDATE CASCADE

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 o*rders* 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.