

DATABASE

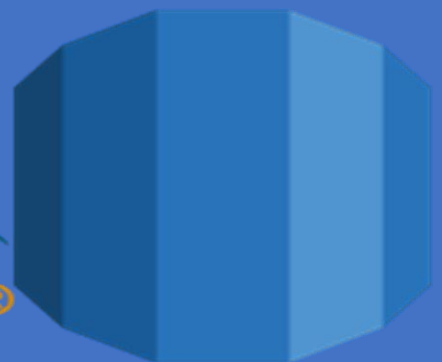
CƠ SỞ DỮ LIỆU

Pisces Kibo



DBeaver

MySQL®



Mục lục

BÀI THỰC HÀNH TUẦN 1	2
BÀI THỰC HÀNH TUẦN 2	16
BÀI THỰC HÀNH TUẦN 3	19
BÀI THỰC HÀNH TUẦN 4	22
BÀI THỰC HÀNH TUẦN 5	24
BÀI THỰC HÀNH TUẦN 6	27
BÀI THỰC HÀNH TUẦN 7	29
BÀI THỰC HÀNH TUẦN 8	31

BÀI THỰC HÀNH TUẦN 1

(Tạo CSDL → Tạo bảng → Primary Key → Foreign Key → Tạo bản ghi → Xóa CSDL)

1. Một số định nghĩa:

- *Database*: một CSDL là một tập hợp các bảng dữ liệu, với dữ liệu có liên quan
- *Table*: bảng dữ liệu: một bảng là một ma trận dữ liệu, một bảng trong CSDL giống như một bảng tính đơn giản
- *Cột*: mỗi cột chứa một kiểu dữ liệu
- *Hàng*: một hàng là một nhóm dữ liệu liên quan
- *Redundancy (dữ liệu dự phòng)*: dữ liệu được lưu giữ hai lần để làm hệ thống nhanh hơn
- *Primary Key (khóa chính – màu vàng)*: là duy nhất. Một giá trị key không thể xuất hiện hai lần trong một bảng. Với một key có thể tìm phần lớn trên một hàng (đối tượng được chỉ đến) → đầu mũi tên hướng đến
- *Foreign Key (khóa ngoại – màu bạc)*: là cái ghim liên kết giữa hai bảng (đối tượng chỉ đến) → đuôi mũi tên hướng đến
- *Compound Key (composite key)*: là một key gồm nhiều cột bởi một cột là không duy nhất
- *Index*: một chỉ mục trong CSDL tương tự như chỉ mục trong một cuốn sách
- *Referential Integrity*: đảm bảo rằng một giá trị Foreign Key luôn trở tới một hàng đang tồn tại

2. Các kiểu dữ liệu:

- **Text**: sử dụng cho các giá trị chuỗi:
 - **TEXT**: sử dụng cho các chuỗi ký tự không bị giới hạn về chiều dài (dùng cho dữ liệu văn bản không xác định)
 - **VARCHAR(N)**: cho phép thực thi các chuỗi có ít hơn N ký tự (nếu không có N sẽ là kiểu text)
 - **CHAR(N)**: thể hiện chuỗi các ký tự với các giá trị được lưu không thay đổi về chiều dài. Nếu một chuỗi nhỏ hơn độ dài cố định N thì sẽ được thêm vào khoảng trắng phía sau (nếu không có N sẽ mặc định là CHAR(1))
- **Numeric**: sử dụng cho các giá trị dữ liệu đại diện cho số lượng và số đo:
 - **Số nguyên**:
 - **TINYINT**: là số nguyên kích cỡ nhỏ từ -128 đến 127 hoặc 0 đến 255
 - **SMALLINT**: là số nguyên có phạm vi từ -32,768 đến 32,767 hoặc 0 đến 4,294,967,295
 - **INTEGER**: là số nguyên có phạm vi từ -2,147,483,648 đến 2,147,483,647
 - **MEDIUMINT**: là số nguyên có kích cỡ trung bình từ -8,388,608 tới 8,388,607 hoặc 0 tới 16,777,215
 - **BIGINT**: là số nguyên kích cỡ lớn từ -9,223,372,036,854,775,808 tới

- 9,223,372,036,854,775,807 hoặc 0 tới 18,446,744,073,709,551,615
- Số thực:
 - FLOAT(M,D): số thực có độ dài hiển thị M và số vị trí sau dấu phẩy D (Phần thập phân có thể lên tới 24 vị trí sau dấu phẩy)
 - DOUBLE(M,D): số thực có độ dài hiển thị M và số vị trí sau dấu phẩy D (Phần thập phân có thể lên tới 53 vị trí sau dấu phẩy)
 - DECIMAL(M,D) = NUMERIC(M,D): mỗi chữ số thập phân chiếm 1 byte. Việc định nghĩa M và D là bắt buộc
- Temporal: sử dụng cho các giá trị dữ liệu đại diện cho ngày và thời gian:
 - DATE: có định dạng YYYY-MM-DD
 - DATETIME: có định dạng YYYY-MM-DD HH:MM:SS
 - TIMESTAMP: có định dạng YYYYMMDDHHMMSS
 - TIME: có định dạng HH:MM:SS
 - YEAR(M): lưu 1 năm trong định dạng 2 chữ số hoặc 4 chữ số (độ dài mặc định là 4)
VD: YEAR(2) → YEAR có thể từ 1970 tới 2069 (70 tới 69). Nếu độ dài được xác định là 4, YEAR có thể từ 1901 tới 2155
- Boolean: sử dụng các giá trị dữ liệu đại diện cho hai trạng thái true/false

3. Các thao tác cơ bản với CSDL:

3.1) Tạo database:

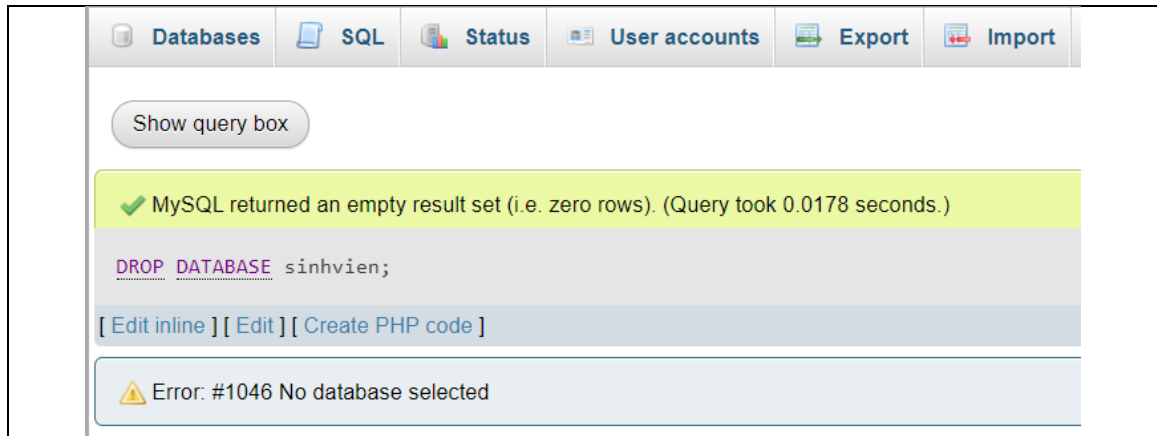
- Cú pháp: **CREATE DATABASE Ten_co_so_du_lieu;**
- VD:



3.2) Xóa database:

- Cú pháp: **DROP DATABASE ten_co_so_du_lieu;**
- VD:

DROP DATABASE sinhvien;

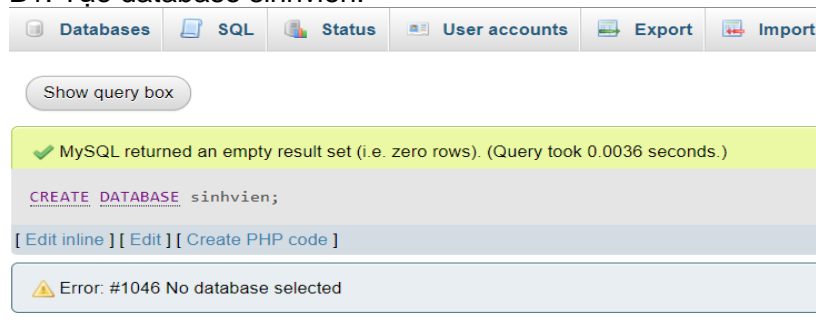


3.3) Tạo bảng:

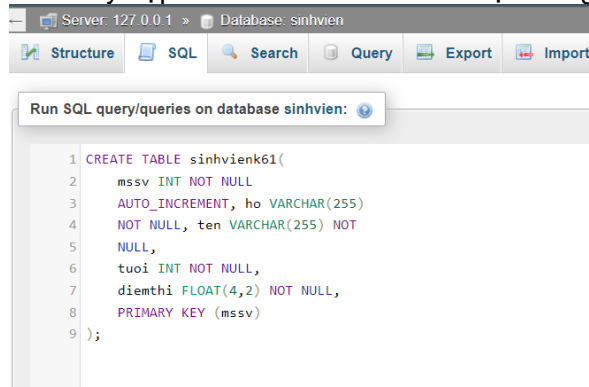
- Cú pháp: ***CREATE TABLE ten_bang (ten_cot kieu_du_lieu);***
- VD:

sinhvienk61(mssv, ho, ten, tuoi, diemthi) trong cơ sở dữ liệu sinhvien:

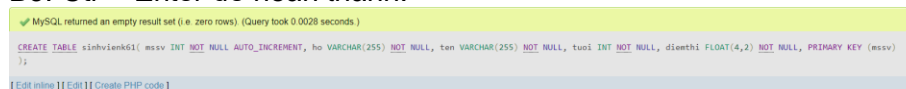
- B1: Tạo database sinhvien:



- B2: Truy cập đến CSDL sinhvien và tạo bảng:



- B3: Ctr + Enter để hoàn thành:



- B4: Hiển thị dữ liệu:

Your SQL query has been executed successfully.

```
SHOW COLUMNS FROM sinhvienk61;
```

☐ Profiling [[Edit inline](#)] [[Edit](#)] [[Create PHP code](#)] [[Refresh](#)]

Extra options

Field	Type	Null	Key	Default	Extra
mssv	int(11)	NO	PRI	NULL	auto_increment
ho	varchar(255)	NO		NULL	
ten	varchar(255)	NO		NULL	
tuoi	int(11)	NO		NULL	
diemthi	float(4,2)	NO		NULL	

Query results operations

[Print](#) [Copy to clipboard](#) [Create view](#)

- Thuộc tính NOT NULL của trường đang được sử dụng bởi vì chúng ta không muốn trường này là NULL
- Thuộc tính AUTO_INCREMENT nói cho MySQL tự động tăng khóa chính và thêm giá trị có sẵn tiếp theo tới trường id.
- Từ khóa PRIMARY KEY được sử dụng để định nghĩa một cột là PRIMARY KEY (khóa chính). Có thể sử dụng nhiều cột phân biệt nhau bởi dấu phẩy để định nghĩa một PRIMARY KEY

3.4) Thay đổi bảng:

- Lệnh ALTER khi muốn thay tên cho một bảng, cho bất kỳ trường nào hoặc nếu bạn muốn thêm hoặc xóa một cột đang tồn tại trong một bảng
- Cú pháp:
 - Xóa cột: **ALTER TABLE nameTable DROP columnsName;**
 - Thêm cột: **ALTER TABLE nameTable ADD columnsName TYPE(N);**

VD:

- B1: Lựa chọn CSDL và tạo bảng hocphik61:

Databases SQL Status User accounts Export Import Settings Replication

Show query box

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0004 seconds.)

```
USE sinhvien;
```

[[Edit inline](#)] [[Edit](#)] [[Create PHP code](#)]

⚠ Error: #1046 No database selected

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0306 seconds.)

```
CREATE TABLE hocphik61 ( ten VARCHAR(40), hocphi INT );
```

[[Edit inline](#)] [[Edit](#)] [[Create PHP code](#)]

- + Lựa chọn CSDL bằng cách sử dụng USE nameCSDL;
- + Tạo bảng cho CSDL đó

- B2: Hiển thị các cột của bảng table trong CSDL:

+ Cú pháp: **SHOW COLUMNS FROM nameTable;**

+ VD:

The screenshot shows the phpMyAdmin interface with the 'Structure' tab selected. A message states: 'Your SQL query has been executed successfully.' Below this, the SQL query is shown: `SHOW COLUMNS FROM hocphik61;`. A table displays the column details:

Field	Type	Null	Key	Default	Extra
ten	varchar(40)	YES		NULL	
hocphi	int(11)	YES		NULL	

Below the table, there are links for 'Profiling', 'Edit inline', 'Edit', 'Create PHP code', and 'Refresh'. At the bottom, there are buttons for 'Print', 'Copy to clipboard', and 'Create view'.

- B3: Xóa thuộc tính tên một cột:
+ Cú pháp: ALTER TABLE nameTable DROP columns;
+ VD:

The screenshot shows the phpMyAdmin interface with the 'Structure' tab selected. A message states: 'MySQL returned an empty result set (i.e. zero rows). (Query took 0.0006 seconds.)' Below this, the SQL query is shown: `ALTER TABLE hocphik61 DROP ten;`. There are links for 'Edit inline', 'Edit', and 'Create PHP code'.

- B4: Thêm thuộc tính tên một cột:
+ Cú pháp: ALTER TABLE nameTable ADD nameCol TYPE(N);
+ VD:

The screenshot shows the phpMyAdmin interface with the 'Structure' tab selected. A message states: 'MySQL returned an empty result set (i.e. zero rows). (Query took 0.0007 seconds.)' Below this, the SQL query is shown: `ALTER TABLE hocphik61 ADD ten VARCHAR(40);`. There are links for 'Edit inline', 'Edit', and 'Create PHP code'.

⇒ Thuộc tính cột mới thêm vào sẽ mặc định ở cuối cùng

- B5: Hiển thị dữ liệu cột:
+ Cú pháp: SHOW COLUMNS FROM nameTable;
+ VD:

Your SQL query has been executed successfully.

```
SHOW COLUMNS FROM hocphik61;
```

☐ Profiling [[Edit inline](#)] [[Edit](#)] [[Create PHP code](#)] [[Refresh](#)]

Extra options

Field	Type	Null	Key	Default	Extra
hocphi	int(11)	YES		NULL	
ten	varchar(40)	YES		NULL	

Query results operations

3.5) Thêm bớt các thuộc tính bảng có thứ tự:

- B1: Xóa thuộc tính tên cột trong bảng:

Show query box

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0006 seconds.)

```
ALTER TABLE hocphik61 DROP ten;
```

[[Edit inline](#)] [[Edit](#)] [[Create PHP code](#)]

⇒ ALTER TABLE hocphik61 DROP ten;

- B2: Thêm thuộc tính vào cột đầu tiên:
 - Cú pháp: **ALTER TABLE nameTable ADD nameColumns TYPE(N) FIRST;**

Show query box

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0006 seconds.)

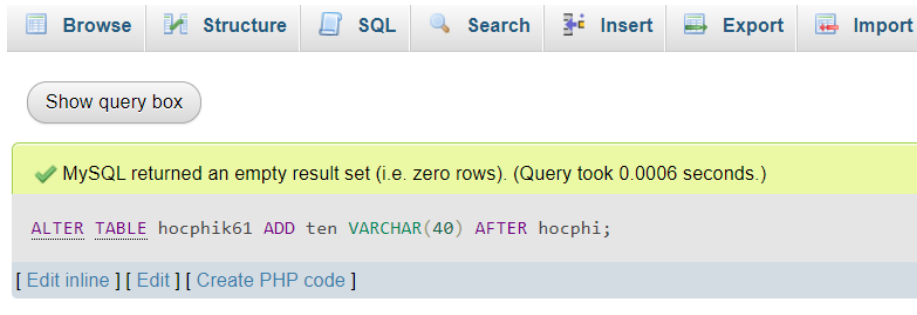
```
ALTER TABLE hocphik61 ADD ten VARCHAR(40) FIRST;
```

[[Edit inline](#)] [[Edit](#)] [[Create PHP code](#)]

⇒ ALTER TABLE hocphik61 ADD ten VARCHAR(40) FIRST;

- B3: Thêm thuộc tính vào cột sau có điều kiện:
 - Cú pháp:

ALTER TABLE nameTable ADD nameColumns TYPE(N) AFTER preColumns;



⇒ ALTER TABLE hocphik61 ADD ten VARCHAR(40) AFTER hocphi;

- Lưu ý:
 - FIRST và AFTER specifier chỉ làm việc với mệnh đề ADD.
 - Điều này nghĩa là nếu muốn tái định vị một cột đang tồn tại bên trong một bảng, đầu tiên bạn phải DROP nó và sau đó ADD nó tại vị trí mới.

3.6) Chuyển đổi kiểu dữ liệu cho thuộc tính:

- Để thay đổi một định nghĩa cột, sử dụng mệnh đề MODIFY hoặc CHANGE cùng với lệnh ALTER
- Cấu trúc:

<i>ALTER TABLE nameTable MODIFY nameColumns TYPE(newValue);</i>
<i>ALTER TABLE nameTable CHANGE oldCol newCol TYPE(newValue);</i>
<i>ALTER TABLE nameTable CHANGE nameCol nameCol TYPE(newValue);</i>

- Ví dụ, để thay đổi cột ten từ VARCHAR(40) thành VARCHAR(20) thì ta sẽ:

<p>Thay đổi kiểu dữ liệu bằng MODIFY</p>	<p>Thay đổi kiểu dữ liệu bằng CHANGE</p>
------------------------------------------	------------------------------------------

Thay đổi kiểu dữ liệu nhưng không thay đổi tên cột bằng CHANGE

3.7) Tác động của ALTER TABLE trên các giá trị NULL và DEFAULT:

- Khi MODIFY hoặc CHANGE một cột, ta có thể xác định cột đó có thể chứa các giá trị NULL và giá trị DEFAULT của nó (nếu không làm điều này sẽ tự động gán các giá trị cho các thuộc tính này)
- Cú pháp:

ALTER TABLE nameTable MODIFY nameColumns DTYPE NOT NULL DEFAULT value;

- VD: cột NOT NULL có giá trị mặc định là 4000000:

ALTER TABLE hocphik61 MODIFY hocphi BIGINT NOT NULL DEFAULT 4000000;

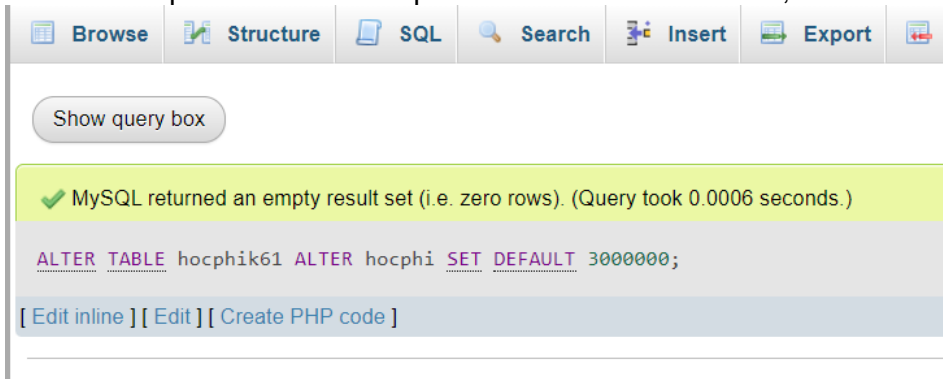
→ Kết quả:

⇒ Nếu không sử dụng lệnh trên thì sẽ mặc định điền các giá trị NULL vào tất cả các cột

3.8) Thay đổi giá trị DEFAULT của một cột:

- Có thể thay đổi một giá trị mặc định cho bất kỳ cột nào bởi sử dụng lệnh ALTER
- Cú pháp thay đổi giá trị DEFAULT:
`ALTER TABLE nameTable ALTER nameColumns SET DEFAULT value;`
- Cú pháp xóa ràng buộc DEFAULT:
`ALTER TABLE nameTable ALTER nameColumns DROP DEFAULT;`
- VD về thay đổi giá trị DEFAULT:

`ALTER TABLE hocphik61 ALTER hocphi SET DEFAULT 3000000;`



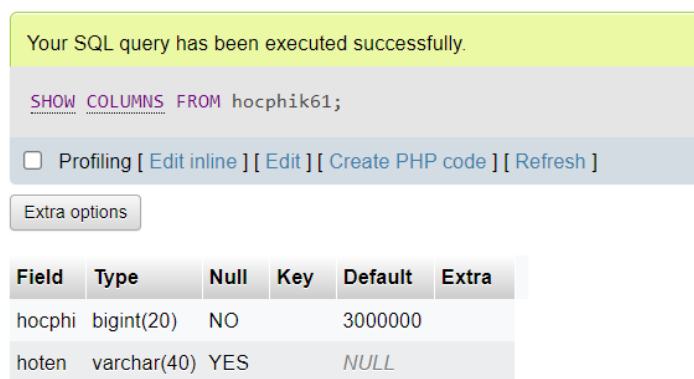
→ Kết quả:

The screenshot shows the 'Table structure' view for the 'hocphik61' table. It has two tabs: 'Table structure' (selected) and 'Relation view'. The table has two columns: 'hocphi' (bigint(20)) and 'hoten' (varchar(40)).

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	hocphi	bigint(20)			No	3000000			Change Drop More
2	hoten	varchar(40)	utf8mb4_general_ci		Yes	NULL			Change Drop More

Below the table, there are options: 'Check all', 'With selected:', 'Browse', 'Change', 'Drop', 'Primary', 'Unique', 'Index', and 'Spatial'.

→ Hiển thị cột:



- VD về xóa giá trị DEFAULT:
`ALTER TABLE hocphik61 ALTER hocphi DROP DEFAULT;`

Show query box

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0006 seconds.)

```
ALTER TABLE hocphik61 ALTER hocphi DROP DEFAULT;
```

[Edit inline] [Edit] [Create PHP code]

→ Kết quả:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/> 1	hocphi	bigint(20)			No	None			Change Drop More
<input type="checkbox"/> 2	hoten	varchar(40)	utf8mb4_general_ci		Yes	NULL			Change Drop More

→ Hiển thị các cột:

Your SQL query has been executed successfully.

```
SHOW COLUMNS FROM hocphik61;
```

☐ Profiling [Edit inline] [Edit] [Create PHP code] [Refresh]

Extra options

Field	Type	Null	Key	Default	Extra
hocphi	bigint(20)	NO		NULL	
hoten	varchar(40)	YES		NULL	

3.9) Thay tên cho bảng:

- Cú pháp: **ALTER TABLE nameOldTable RENAME TO nameNewTable;**
- VD:

Browse Structure SQL Search Insert Export Import

Show query box

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0007 seconds.)

```
ALTER TABLE hocphik61 RENAME TO hocphik62;
```

[Edit inline] [Edit] [Create PHP code]

3.10) Xóa bảng:

- Cú pháp: **DROP TABLE ten_bang;**

- VD:

Show query box

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0006 seconds.)

```
DROP TABLE sinhvienk61;
```

[Edit inline] [Edit] [Create PHP code]

3.11) Chèn dữ liệu vào bảng:

- Cú pháp:

***INSERT INTO ten_bang (truong1, truong2,...truongN)
VALUES(giatri1, giatri2,...giatriN);***

- VD:

- B1: Tạo bảng mới:

Show query box

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0007 seconds.)

```
CREATE TABLE sinhvienk61 ( ho VARCHAR(10), ten VARCHAR(10), diemthi FLOAT(4,2) );
```

[Edit inline] [Edit] [Create PHP code]
- B2: Hiện thị cột mới vừa tạo:

Your SQL query has been executed successfully.

```
SHOW COLUMNS FROM sinhvienk61;
```

☐ Profiling [Edit inline] [Edit] [Create PHP code] [Refresh]

Extra options

Field	Type	Null	Key	Default	Extra
ho	varchar(10)	YES		NULL	
ten	varchar(10)	YES		NULL	
diemthi	float(4,2)	YES		NULL	
- B3: Chèn các giá trị dữ liệu vào trong thuộc tính:

Show query box

✓ 1 row inserted. (Query took 0.0006 seconds.)

```
INSERT INTO sinhvienk61 (ho, ten, diemthi) VALUES ("Dinh Van", "Cao", 8);
```

[Edit inline] [Edit] [Create PHP code]

✓ 1 row inserted. (Query took 0.0027 seconds.)

```
INSERT INTO sinhvienk61 (ho, ten, diemthi) VALUES ("Nguyen Van", "Thanh", 9);
```

[Edit inline] [Edit] [Create PHP code]

✓ 1 row inserted. (Query took 0.0006 seconds.)

```
INSERT INTO sinhvienk61 (ho, ten, diemthi) VALUES ("Nguyen Hoang", "Manh", 7.5);
```

[Edit inline] [Edit] [Create PHP code]

⚠ Warning: #1265 Data truncated for column 'ho' at row 1

✓ 1 row inserted. (Query took 0.0005 seconds.)

```
INSERT INTO sinhvienk61 (ho, ten, diemthi) VALUES ("Tran Van", "Nam", 10);
```

[Edit inline] [Edit] [Create PHP code]

- B4: Xem dữ liệu:

```
Select * from sinhvienK61;
```

☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

☐ Show all | Number of rows: 25 | Filter rows: Search this table

Extra options

ho	ten	diemthi
Dinh Van	Cao	8.00
Nguyen Van	Thanh	9.00
Nguyen Hoa	Manh	7.50
Tran Van	Nam	10.00

3.12) Cập nhật dữ liệu trong bảng:

- Dữ liệu đang tồn tại trong một bảng có thể cần được sửa đổi. Ta có thể thực hiện điều này bởi sử dụng lệnh UPDATE trong SQL. Lệnh này sẽ sửa đổi bất kỳ giá trị trường nào trong bất cứ bảng MySQL nào
 - Có thể cập nhật một hoặc nhiều trường
 - Có thể xác định bất kỳ điều kiện nào bởi sử dụng mệnh đề WHERE
 - Có thể cập nhật các giá trị trong một bảng đơn tại một thời điểm
 - Mệnh đề WHERE là hữu ích khi bạn muốn cập nhật các hàng đã chọn trong một bảng
- Cú pháp UPDATE giá trị với điều kiện có sẵn:

UPDATE ten_bang SET nameColumn1 = newValue, nameColumn2 = newValue2 [WHERE điều kiện];

- Cú pháp DELETE dữ liệu trong bảng:

DELETE FROM ten_bang WHERE điều_kiện;

- Nếu mệnh đề WHERE không được xác định, thì tất cả bản ghi sẽ bị xóa từ bảng MySQL đã cho
- Có thể xác định bất kỳ điều kiện nào với mệnh đề WHERE
- Có thể xóa các bản ghi trong một bảng đơn tại cùng một thời điểm
- Mệnh đề WHERE là thực sự hữu ích khi bạn muốn xóa các hàng đã chọn trong một bảng. Nếu không xác định mệnh đề WHERE thì thật sự rất nguy hiểm bởi vì toàn bộ bảng của bạn sẽ bị xóa

• VD:

```
UPDATE sinhvienk61 SET ten="Huong" WHERE diemthi = 8;
```

✓ 1 row affected. (Query took 0.0016 seconds.)

```
UPDATE sinhvienk61 SET ten="Huong" WHERE diemthi=8;
```

[Edit inline] [Edit] [Create PHP code]

⇒ Cập nhật lại 'ten' với điều kiện 'diemthi' đã biết

→ Xóa dữ liệu bảng

✓ 0 rows affected. (Query took 0.0008 seconds.)

```
DELETE FROM sinhvienk61 WHERE diemthi=3;
```

[Edit inline] [Edit] [Create PHP code]

⇒ Xóa dữ liệu tại vị trí "diemthi = 3"

3.13) Nếu trùng ghi đè cái mới: (xóa cũ ghi mới)

• Cấu trúc:

```
REPLACE INSERT INTO nameDatabase (  
    nameCol1, nameCol2, ...  
) VALUES (  
    Val1, Val2, ...  
);
```

VD:

```
replace insert into classicmodels.offices (  
    officeCode, city, phone, addressLine1, country, postalCode, territory  
) values (  
    10, 'ThanhHoa', '+84398463203', 'addition2', 'VN', '80000', 'VN');
```

3.14) Nếu tồn tại thì không thực hiện chức năng:

- Cấu trúc:

```
INSERT IGNORE INTO nameDatabases (  
    nameCol1, nameCol2, nameCol3, ...  
) VALUES (  
    Value1, Value2, Value3, ...);
```

VD:

```
insert ignore into classicmodels.offices (  
    officeCode, city, phone, addressLine1, country, postalCode, territory  
) values (  
    10, 'ThanhHoa', '+84398463203', 'addition2', 'VN', '80000', 'VN');
```


BÀI THỰC HÀNH TUẦN 2

KHÓA CHÍNH VÀ KHÓA NGOẠI

1. Khóa ngoại (Foreign Key – màu bạc):

- Foreign key là mối quan hệ giữa hai bảng (gọi là cha con), nghĩa là nếu bảng A có một thuộc tính liên kết tới bảng B thì lúc này bảng B đóng vai trò là cha và bảng A đóng vai trò là con
- Khái niệm có ở MySQL, hệ quản trị CSDL (SQL Server, Oracle, Access, ...)
- Có hai loại khóa ngoại: khóa ngoại liên kết giữa hai bảng và khóa ngoại trỏ đến chính nó (đệ quy)
- So sánh hai loại khóa ngoại

Khóa ngoại giữa hai bảng	Khóa ngoại trỏ đến chính bảng đó
Khóa ngoại ở bảng A sẽ tham chiếu đến khóa chính của bảng B (bảng B là cha và bảng A là con)	Trong bảng table, khóa ngoại A trỏ đến khóa chính B trong cùng một bảng table (sơ đồ này gọi là đệ quy – khóa ngoại sẽ tham chiếu tới chính table)

2. Tạo khóa ngoại Foreign Key:

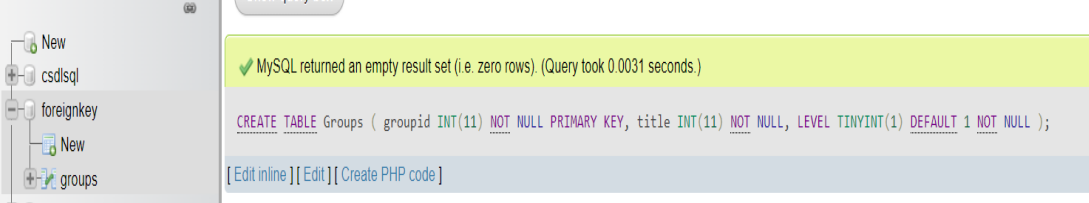
a) Tạo khóa ngoại bằng lệnh tạo bảng create table:

- Cấu trúc:

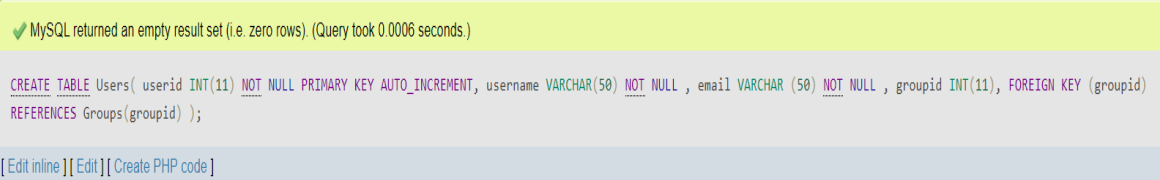
```
CREATE TABLE nameTable (  
    nameColA TYPE(n) NOT NULL PRIMARY KEY,  
    nameColB TYPE(n) NOT NULL  
);
```

VD:

+ B1: Tạo bảng Groups không có PRIMARY KEY



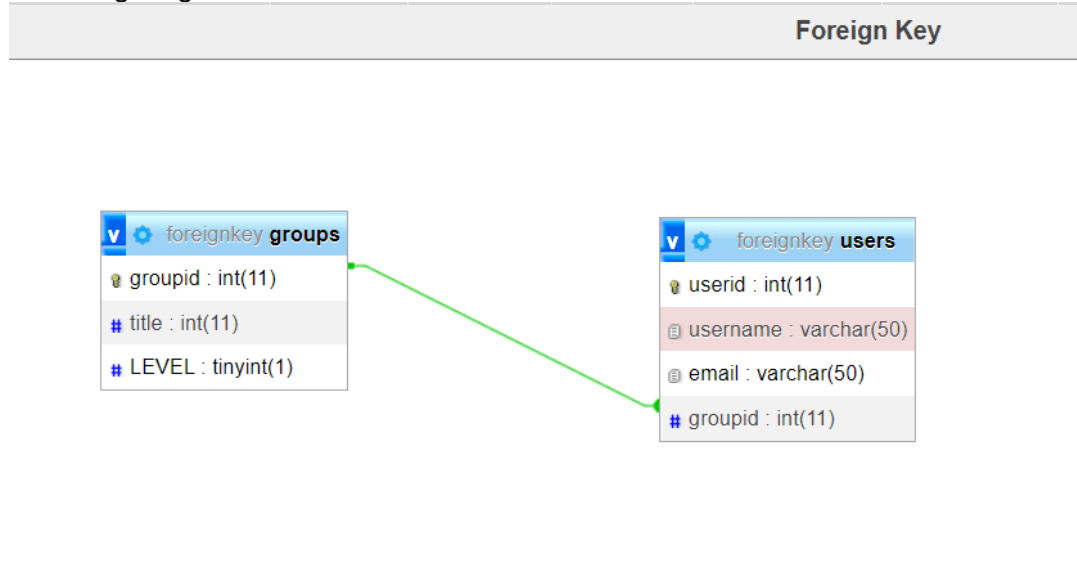
+ B2: Tạo bảng Users



⇒ FOREIGN KEY (groupid) là field chọn làm khóa ngoại ở bảng con, tức bảng Users

⇒ REFERENCES Groups(groupid) là khóa chính của bảng cha, tức bảng groups

B3: Bảng diagram



- Cách tạo khóa chính và khóa phụ bằng SQL:

Khóa ngoại (Foreign)	Khóa chính (References)
<pre>CREATE TABLE tableB(CONSTRAINTS nameFK FOREIGN KEY (columnsB) REFERENCES tableA(columnsA) check <điều kiện>);</pre>	

- Đặt tên cho khóa ngoại bằng từ khóa CONSTRAINTS

b) Tạo khóa ngoại bằng ALTER TABLE:

- Tạo trước hai bảng A và B không có khóa:

VD:

```
CREATE TABLE Groups (
  groupid INT(11) NOT NULL PRIMARY KEY,
  title INT(11) NOT NULL,
  LEVEL TINYINT(1) DEFAULT 1 NOT NULL
);
CREATE TABLE Users(
  userid INT(11) NOT NULL PRIMARY KEY AUTO_INCREMENT,
  username VARCHAR(50) NOT NULL ,
  email VARCHAR (50) NOT NULL ,
  groupid INT(11)
);
ALTER TABLE Users ADD FOREIGN KEY(groupid) REFERENCES Groups(groupid);
```

Hoặc

```
ALTER TABLE Users ADD CONSTRAINT fk_group FOREIGN KEY(groupid) REFERENCES Groups(groupid);
```

- Sau đó xử dụng câu lệnh:
 - Loại 1: không đặt tên:

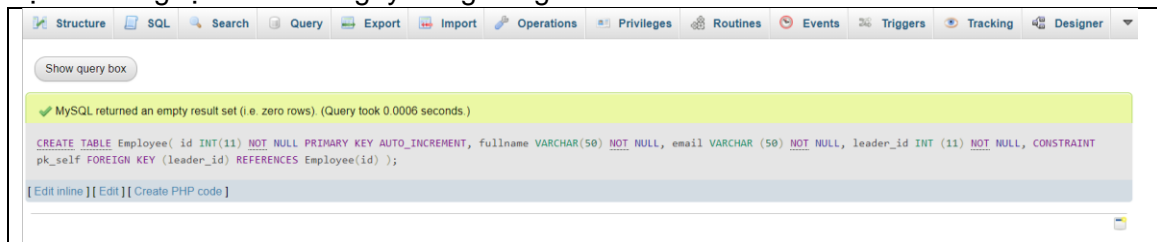
```
ALTER TABLE tableB ADD FOREIGN KEY(columnsB) REFERENCES tableA(columnsA);
```

- Loại 2: có đặt tên

```
ALTER TABLE tableB ADD CONSTRAINT fk_name FOREIGN KEY(columnsB) REFERENCES tableA(coumnsA);
```

c) Tạo khóa ngoại trong trường hợp tham chiếu chính nó:

- Tạo khóa ngoại chính nó ngay trong bảng:



- Cấu trúc:

```
CREATE TABLE newTable(  
    CONSTRAINT pk_self FOREIGN KEY (id1) REFERENCES newTable(id2)  
);
```

3. Xóa (Drop) Foreign Key:

- Cú pháp: **ALTER TABLE tableName DROP FOREIGN KEY fk_name;**
- Lưu ý: chỉ thực hiện được thao tác xóa khi không tồn tại một bảng con nào tham chiếu đến nó

BÀI THỰC HÀNH TUẦN 3

TRUY VẤN ĐƠN GIẢN

0. Chú thích:

-- <i>Chú thích 1 dòng</i>	/* <i>Chú thích nhiều dòng</i> */
----------------------------	-----------------------------------------

1. Tổng quát:

- + SELECT: xác định các cột cần đưa ra kết quả
- + FROM: xác định các bảng cần lấy thông tin
- + WHERE: xác định các điều kiện
 - ⇒ Khối lệnh SELECT – FROM – WHERE được bổ sung thêm các mệnh đề GROUP BY, HAVING, ORDER BY và các hàm hỗ trợ tính toán MAX, MIN, COUNT, SUM, AVG
- + Cú pháp:

```
SELECT [tính chất] <danh sách các thuộc tính_1>  
FROM <danh sách các table hoặc query/view [as alias] >  
[WHERE <điều kiện_1>]  
[GROUP BY <danh sách các thuộc tính_2>]  
[HAVING <điều kiện_2>]  
[ORDER BY <danh sách các thuộc tính_3 [ASC | DESC]>]  
LIMIT <giới hạn>
```

- [tính chất]:
 - ALL: chọn ra tất cả các dòng trong bảng
 - DISTINCT: loại bỏ các cột trùng lặp thông tin
 - DISTINCTROW: loại bỏ các dòng trùng lặp thông tin
 - TOP<n>: chọn n dòng đầu tiên thỏa mãn điều kiện
- <danh sách các thuộc tính_1>:
 - Tên các thuộc tính cách nhau bởi dấu ,
 - Lấy tất cả thuộc tính của bảng dùng *
 - Nếu có nhiều bảng thì ghi nameTable1.thuoc tinhA, nameTableB.thuoc tinhB, ...
- <danh sách các bảng>:
 - Các table được để trong khối lệnh này (table1, table2, ...)
 - Nếu muốn đặt tên cho bảng để dễ phân biệt thì sử dụng từ khóa “AS” (nameTable AS tableA, nameTable AS tableB, ...)
- <điều kiện 1> là để lọc dữ liệu chính

- <danh sách các thuộc tính_2>: là các dữ liệu được gom nhóm lại theo cột này (ưu tiên từ trái sang phải)
- <điều kiện 2>: điều kiện lọc lại dữ liệu sau khi đã thực hiện tính toán trên dữ liệu. Điều kiện này được áp dụng trên dữ liệu thỏa mãn điều kiện_1
- <danh sách các thuộc tính_3>: là dùng để sắp xếp thứ tự ưu tiên từ trái qua phải (ASC – tăng, DESC – giảm), mặc định là tăng dần

2. Truy vấn đơn giản:

+ Cú pháp: SELECT <dstt> FROM <nameTable>;

a) Tìm kiếm có sắp xếp ưu tiên từ trái qua phải:

+ Cú pháp:

```
SELECT ...
FROM ...
ORDER BY thuộc_tính_1[ASC|DESC], thuộc_tính_2[ASC|DESC], ...
```

- Đầu tiên xếp theo <nameCol1> trước, nếu <nameCol1> có giá trị ngang nhau thì sẽ xếp theo <nameCol2>

b) Tìm kiếm với điều kiện đơn giản:

- AND và OR: (và, hoặc)

```
SELECT ...
FROM ...
WHERE (điều_kiện_1) AND/OR .... (điều_kiện_n)
```

- BETWEEN ... AND và NOT BETWEEN ... AND: (nằm giữa khoảng)

```
SELECT *
FROM nameTable
WHERE Columns BETWEEN valueStart AND valueEnd;
```

- IS NULL và IS NOT NULL: (kiểm tra giá trị rỗng)

```
SELECT *
FROM nameTable
WHERE ColumnName IS NULL (IS NOT NULL);
```

- IN và NOT IN: (kiểm tra có các giá trị nằm trên tập hợp)

```
SELECT nameCol
FROM nameTable
WHERE nameColumns IN ('H1','H2','H3', ...)
```

c) Tìm kiếm có xử lý chuỗi ký tự:

- So sánh giữa các chuỗi ký tự sử dụng từ khóa “LIKE”
 - “%String”: tất cả các từ đứng trước String
 - “String%”: tất cả các từ khóa đứng sau String
 - “_”: thay thế cho ký tự bất kỳ nào đó
 - Toán tử LIKE = Contain - Starswith% - %Endswith
 - Chú ý:
 - Like “ab\%cd%” cho ra những chuỗi bắt đầu với “ab%cd”
 - Like “ab\cd%” cho ra những chuỗi bắt đầu với “ab\cd”
- ⇒ Ta sử dụng dấu “escape(\\)” đứng trước các ký tự đặc biệt như %_

d) Tìm kiếm có điều kiện liên quan đến ngày tháng:

- Hàm DATE(expr): trích một phần date từ biểu thức expr của date hoặc datetime
→ DATE(‘yyyy-mm-dd hh:mm:ss’)
- Hàm DATEDIFF(expr1, expr2): trả về expr1 – expr2 được biểu diễn dưới dạng số ngày từ một date tới date khác
→ DATEDIFF(‘yyyy-mm-dd hh:mm:ss’, ‘yyyy-mm-dd’)

Function	Mô tả
NOW()	Trả về ngày tháng và thời gian hiện tại
CURDATE()	Trả về ngày hiện tại
CURTIME()	Trả về thời gian hiện tại
DATE()	Trích xuất các phần ngày của một ngày hoặc biểu thức ngày/ thời gian
EXTRACT()	Trả về một phần của ngày tháng
DATE_ADD()	Thêm một khoảng thời gian nhất định vào một ngày
DATE_SUB()	Bớt một khoảng thời gian nhất định từ một ngày
DATEDIFF()	Trả về số lượng ngày giữa hai khoảng ngày tháng nào đó
DATE_FORMAT()	Hiển thị dữ liệu ngày tháng/thời gian trong các định dạng khác

BÀI THỰC HÀNH TUẦN 4

PHÉP KẾT TRONG SQL

1. Phép kết trong tìm kiếm nhiều bảng:

- INNER JOIN hoặc JOIN: phép kết giữa hai bảng A và B là bảng $C = \{A, B\}$ sao cho thỏa mãn điều kiện sau từ khóa "ON"

```
SELECT * FROM tableA, tableB ON tableA.column = tableB.column;  
SELECT * FROM tableA JOIN tableB ON tableA.column = tableB.column;  
SELECT * FROM tableA INNER JOIN tableB ON tableA.column = tableB.column;
```

- RIGHT JOINS (kết phải): phép kết RIGHT OUTER JOINS giữa hai bảng A và B là bảng $C = \{A, B\} + \{\text{các bộ còn lại trong B mà không thỏa mãn điều kiện với bất kỳ điều kiện nào trong A} \rightarrow \text{NULL}\}$

```
SELECT * FROM tableA RIGHT JOIN tableB ON tableA.column = tableB.column;
```

- LEFT JOINS (kết trái): phép kết LEFT OUTER JOINS giữa hai bảng A và B là bảng $C = \{A, B\} + \{\text{các bộ còn lại trong A mà không thỏa mãn điều kiện với bất kỳ điều kiện nào trong B} \rightarrow \text{NULL}\}$

```
SELECT * FROM tableA LEFT JOIN tableB ON tableA.column = tableB.column;
```

- FULL JOINS = RIGHT JOINS + LEFT JOINS

➔ Phép kết FULL OUTER JOINS giữa hai bảng A và B là bảng $C = \{A, B\} + \{\text{các bộ còn lại trong B mà không thỏa mãn điều kiện với bất kỳ điều kiện nào trong A} \rightarrow \text{NULL}\} + \{\text{các bộ còn lại trong A mà không thỏa mãn điều kiện với bất kỳ điều kiện nào trong B} \rightarrow \text{NULL}\}$

```
SELECT * FROM tableA FULL JOIN tableB ON tableA.column = tableB.column;
```

2. Câu truy vấn sử dụng GROUP BY:

+ Dùng để gom nhóm dữ liệu, thường dùng kết hợp với một hàm tính toán kể trên

+ Cú pháp: NameTable(col1, col2, col3, ...)

```
SELECT col1, <Mathtype> (col2), ...  
FROM nameTable  
GROUP BY col3, ...
```

VD: cho biết mức lương lớn nhất trong từng phòng ban:

```
SELECT PHG, max(LUONG)  
FROM nameTable  
GROUP BY PHG
```

3. Mệnh đề điều kiện HAVING:

+ Thường sử dụng cùng với mệnh đề GROUP BY (biểu thức điều kiện sau HAVING không tác động vào toàn bảng từ FROM mà chỉ tác động lần lượt từng nhóm các mẫu tin đã chỉ ra trong mệnh đề GROUP BY)

+ Cú pháp:

```
SELECT col1, <Mathtype> (col2), ...  
FROM nameTable  
GROUP BY col3, ...  
HAVING <điều kiện cho GROUP BY>
```


BÀI THỰC HÀNH TUẦN 5

TRUY VẤN LÒNG TRONG SQL

1. Tìm kiếm có lượng từ EXISTS, ANY, ALL:

+ Tìm danh sách A biết có ít nhất 1 B (EXISTS)

Cách 1	Cách 2
<pre>SELECT colA1, colA2, ... FROM tableA WHERE EXISTS (SELECT colB, colB, ... FROM tableB WHERE tableB.columnE = tableA.columnE);</pre>	<pre>SELECT colA, colB, ... FROM tableA WHERE (SELECT COUNT(*) FROM tableB WHERE tableB.colE = tableA.colE) > 0;</pre>

- Chú ý: ANY tương đương với toán tử IN

+ Tìm đối tượng có Columns lớn nhất

Cách 1	Cách 2
<pre>SELECT colA, colB FROM nameTable WHERE colB >= ALL (SELECT colB FROM nameTable);</pre>	<pre>SELECT colA, colB FROM nameTable WHERE colB = (SELECT MAX(colB) FROM nameTable);</pre>

2. Truy vấn lồng phân cấp:

- Mệnh đề WHERE của truy vấn con không tham chiếu đến thuộc tính của các quan hệ trong mệnh đề FROM của truy vấn cha
- Khi thực hiện, truy vấn con luôn được thực hiện trước

VD1: Cho biết các đối tượng cùng nameColumns với nhân viên “Tạ Quang Tùng”

```
SELECT colA, colB
FROM nameTable
WHERE nameColumns IN (
    SELECT nameColumns
    FROM nameTable
    WHERE colB = “Tạ Quang Tùng”);
```

VD2: Tìm những đối tượng có nameColumns lớn hơn nameColumns của tất cả đối tượng trong phòng 3

```

SELECT colA, colB
FROM nameTable
WHERE nameColumns > (
    SELECT MAX(nameColumns)
    FROM nameTable
    WHERE PHG = 3);

```

VD3: Tìm phòng ban có đông nhân viên nhất (gom nhóm + truy vấn lồng phân cấp)

```

SELECT pb.TENPHG, COUNT(*) AS SOLUONG
FROM NHANVIEN nv, PHONGBAN pb
WHERE nv.PHG = pb.PHG
GROUP BY nv.PHG, pb.TENPHG
HAVING COUNT(*) >= ALL(
    SELECT COUNT(*)
    FROM NHANVIEN
    GROUP BY PHG);

```

VD4: Tìm đối tượng có nameColumns lớn hơn mức nameColumns của một đối tượng nào đó trong phòng “Nghiên cứu”

```

SELECT colA, colB, colC, PHG
FROM nameTable1
WHERE nameColumns > any(
    SELECT nameTable1 nv, nameTable2 pb
    WHERE nv.PHG = pb.MAPHG AND pb.TENPHG = 'Nghiên cứu');

```

3. Truy vấn lồng tương quan:

- Mệnh đề WHERE của truy vấn con tham chiếu ít nhất một thuộc tính của các quan hệ trong mệnh đề FROM ở truy vấn cha
- Khi thực hiện, câu truy vấn con sẽ được thực hiện nhiều lần, mỗi lần ứng với một bộ của truy vấn cha

VD1: Tìm những đối tượng không có con nào

```

SELECT colA, colB
FROM nameTableCha n
WHERE NOT EXISTS (
    SELECT *
    FROM nameTableCon t
    WHERE t.MANV = n.MANV);

```

VD2: Tìm tất cả các đối tượng ở phòng “Nghiên cứu”

```

SELECT colA, colB
FROM tableA n

```

```
WHERE EXISTS (  
    SELECT *  
    FROM tableB p  
    WHERE TENPHG = 'Nghiên cứu' AND p.MAPHG = n.PHG);
```

BÀI THỰC HÀNH TUẦN 6

PHÉP CHIA TRONG SQL

1. Các dạng phép chia:

+ Cách 1: Sử dụng NOT EXISTS + NOT IN hoặc NOT EXISTS + NOT EXISTS

// Tìm những đối tượng A được <đối tượng C> trong tất cả các <đối tượng B>

■ Cách 1:

```
SELECT tab1.colA, tab1.colB, ...
FROM nameTable1 AS tab1
WHERE NOT EXISTS (
    SELECT *
    FROM nameTable2 AS tab2, nameTable3 AS tab3, ...
    WHERE tab2.colC = tab3.colC AND <điều kiện khác>
    AND tab3.colD NOT IN
    (
        SELECT tab4.colD
        FROM nameTable4 AS tab4
        WHERE tab4.colE = tab1.colE
        ...
    )
);
```

■ Cách 2:

```
SELECT tab1.colA, tab1.colB, ...
FROM nameTable1 AS tab1
WHERE NOT EXISTS (
    SELECT *
    FROM nameTable2 AS tab2, nameTable3 AS tab3, ...
    WHERE tab2.colC = tab3.colC AND <điều kiện khác>
    AND NOT EXISTS
    (
        SELECT *
        FROM nameTable4 AS tab4
        WHERE tab4.colE = tab1.colE
        ...
    )
);
```

+ Cách 2: Sử dụng mệnh đề GROUP BY + HAVING

// Tìm những đối tượng A được <đối tượng C> trong tất cả các <đối tượng B>

```
SELECT tab1.colA, tab1.colB, ...
FROM nameTable1 tab1, nameTable2 tab2, nameTable3 tab3, ...
WHERE <góm nhóm lại các điều kiện với nhau>
```

GROUP BY <từ mỗi thuộc tính>
HAVING <các điều kiện khác (bao gồm truy vấn lồng)>;

- R là đối tượng B
- S là đối tượng C
- R/S = <kết quả>

2. Các loại truy vấn khác:

- Truy vấn con ở mệnh đề SELECT:

```
SELECT colA, colB, <truy vấn con> AS nameColumns  
FROM nameTable tableName;
```

với <truy vấn con> gồm:
(select thuocTinh from nameTable2
where nameTable2.colD = tableName.colD)

- Truy vấn con ở mệnh đề FROM:

+ Kết quả trả về của một câu truy vấn phụ là một bảng (bảng trung gian và không lưu trữ thật sự)

```
SELECT colA, colB, ...  
FROM nameTable tableName, <truy vấn con> AS nameColumns;
```

với <truy vấn con> gồm:
(select thuocTinh from nameTable2
where nameTable2.colD = tableName.colD)

- Điều kiện kết ở mệnh đề FROM: (sử dụng cấu trúc JOIN, INNER JOIN, LEFT JOIN, RIGHT JOIN, ...)
- Cấu trúc CASE (câu lệnh điều kiện):

```
CASE <biethuc_dauvao>  
WHEN biethuc_1 THEN ketqua_1  
WHEN biethuc_2 THEN ketqua_2  
...  
WHEN biethuc_n THEN ketqua_n  
ELSE ketqua_khac  
END
```

BÀI THỰC HÀNH TUẦN 7

LẬP TRÌNH CƠ SỞ DỮ LIỆU TRUY VẤN SQL

1. Tạo MySQL Stored Procedure đầu tiên:

+ Cấu trúc:

```
DELIMITER $$
CREATE `NAMEDATABASES` fuctionActive()
BEGIN
    /* Các hoạt động thực thi */
END; $$
DELIMITER;
```

- DELIMITER \$\$: phân cách bộ nhớ lưu trữ thủ tục Cache để mở ra ô lưu trữ mới
- **CREATE NAMEDATABASES** fuctionActive() : khai báo CSDL mới với fuctionActive là tên thủ tục
- BEGIN và END; \$\$: khai báo bắt đầu và kết thúc của DATABASES
- Đóng lại ô lưu trữ DELIMITER; → giới hạn khoảng truy vấn

a) Gọi Stored Procedure: (gọi hàm)

- Cấu trúc: CALL storeName();

b) Xem danh sách Stored Procedure trong hệ thống:

- Cấu trúc: show nameDatabase status;

c) Sửa Stored Procedure đã tạo: (sửa hàm)

```
DELIMITER $$

DROP nameDatabases IF EXISTS functionActive()
BEGIN
    SELECT * FROM nameTable;
END$$

DELIMITER;
```

- ⇒ Trong CSDL không cung cấp lệnh sửa Stored nên sẽ chạy lệnh tạo mới
- ⇒ Phải dùng lệnh Drop để xóa database rồi tạo lại ngay trong cùng DELIMITER;

2. Biến (variable) trong MySQL Stored Procedure:

a) Khai báo biến:

- Cú pháp:
DECLARE variableName <datatype (size)> DEFAULT defaultValue
 - DECLARE: từ khóa tạo biến
 - variableName: tên biến
 - datatype(size): kiểu dữ liệu và kích cỡ của nó

- DEFAULT defaultValue: gán giá trị mặc định cho biến

b) Gán giá trị cho biến:

- Cú pháp:
SET variableName = 'value';
- Gán giá trị cho biến bằng SELECT INTO:

```
DECLARE newNameValue INT DEFAULT 0  
SELECT <điều kiện thực hiện> INTO newNameValue  
FROM nameTable
```

BÀI THỰC HÀNH TUẦN 8

SO SÁNH SQL VÀ NOSQL

<u>SQL</u>	<u>NoSQL</u>
Sử dụng hệ quản trị CSDL quan hệ	Sử dụng hệ thống CSDL phân tán
Phù hợp với các truy vấn phức tạp	Phù hợp với các truy vấn ít phức tạp hơn
Tiêu chuẩn: <ul style="list-style-type: none">• Tính nguyên tử (Atomicity)• Tính nhất quán (Consistency)• Tính cô lập (Isolation)• Tính bền vững (Durability)	Tiêu chuẩn: <ul style="list-style-type: none">• Tính nhất quán (Consistency)• Availability• Partition tolerance