Khoa CNTT
Đại Học Đà Lạt

HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU

PHẦN II:

CƠ SỞ DỮ LIỆU MY SQL

Th.s. Đoàn Minh Khuê
khuedm@dlu.edu.vn

Nội dung trình bày

- Tổng Quan
- Bảng
- Toán Tử
- Phát biểu SQL
- Import và Export dữ liệu

Tổng Quan

- **Giới thiệu cơ sở dữ liệu**
- **Cơ sở dữ liệu My SQL**

Giới thiệu cơ sở dữ liệu

- **Khái niệm**
- **Chức năng**
- **Các loại CSDL**
- **Các đối tượng chính của CSDL quan hệ**
- **Hệ quản trị CSDL**
- **SQL (Structure Query Language)**

Khái niệm

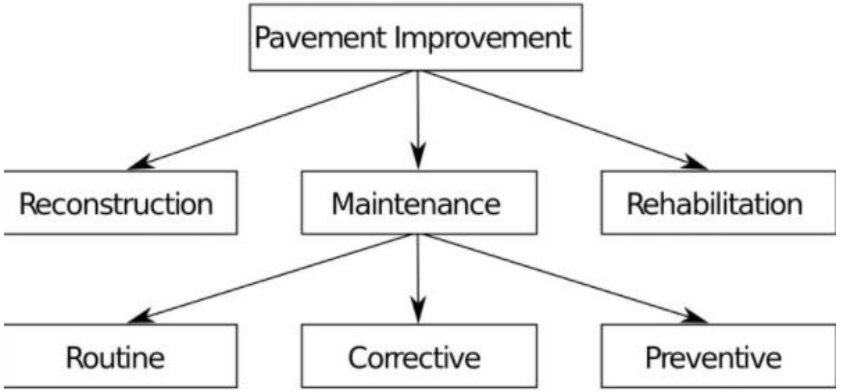
- CSDL là một tập hợp dữ liệu được lưu trữ một cách có tổ chức nhằm giúp việc xem, tìm kiếm và lấy thông tin được nhanh chóng và chính xác, giúp giảm công sức và thời gian quản lý thông tin cần thiết

Chức năng CSDL

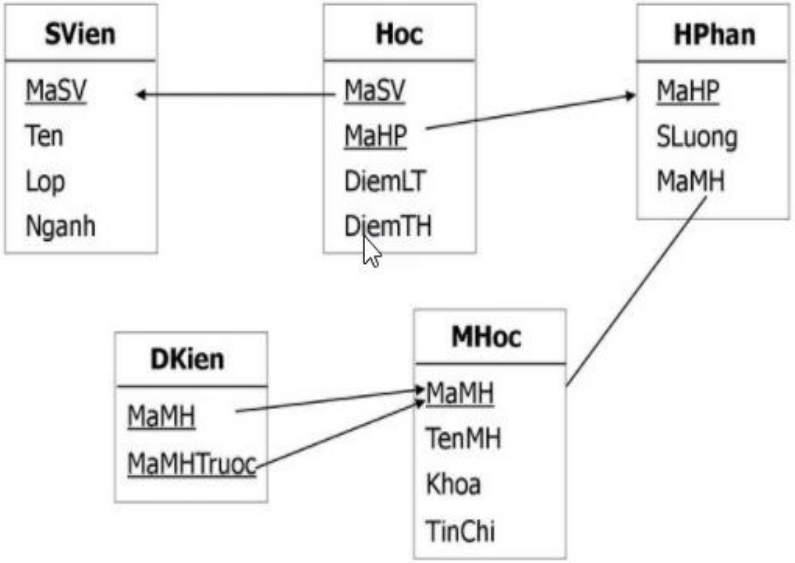
- Lưu trữ
- Truy cập
- Tổ chức
- Xử lý

Các loại CSDL

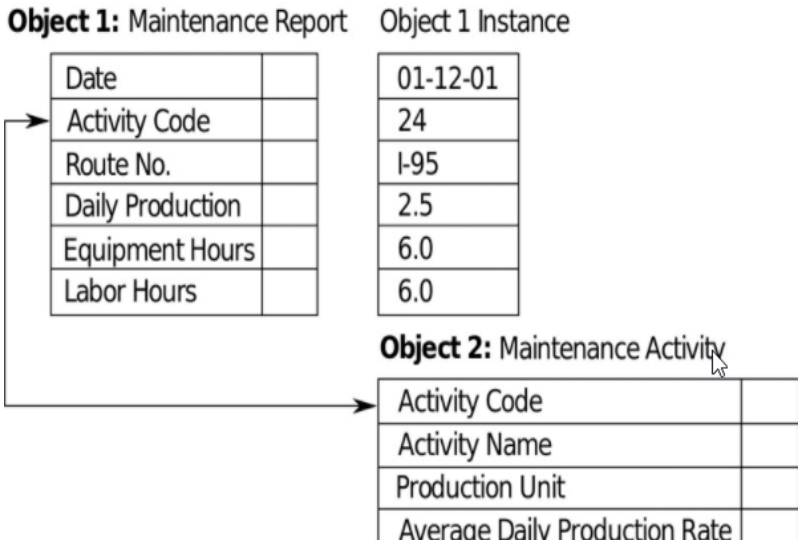
- CSDL phân cấp
- CSDL hướng đối tượng
- CSDL quan hệ



Mô hình CSDL phân cấp



Mô hình CSDL quan hệ



Mô hình CSDL hướng đối tượng

Các đối tượng chính của CSDL Quan hệ

- **Bảng dữ liệu (table)**
- **Các Quan hệ**

Bảng dữ liệu

- Là thành phần trung tâm của CSDL, được dùng để lưu trữ thông tin của CSDL
- Gồm hai thành phần: dòng và cột
  - **Cột**: là một khối dữ liệu trong bảng, có cùng loại dữ liệu, có các thông tin chính:
    - Tên cột: dùng để phân biệt với các cột khác trong bảng. Tên cột trong bảng phải duy nhất và không dùng các ký tự đặc biệt.
    - Kiểu dữ liệu của cột: xác định loại giá trị nào được phép lưu trữ trong cột

MKH	TEN_KH	PHAI	DIA_CHI	DT	EMAIL
KH001	Trần Văn An	0	123 Nguyễn Du	8123456	tvan@yahoo.com
KH002	Nguyễn Thanh An	0	30 Lê Thánh Tôn	9852147	ntan@yahoo.com
KH003	Lê Thanh Thảo	1	22bis Pasteur	8976431	ltthao@gmail.com

Bảng dữ liệu

- Là thành phần trung tâm của CSDL, được dùng để lưu trữ thông tin của CSDL
- Gồm hai thành phần: dòng và cột
  - **Dòng**: là tập hợp các thông tin của tất cả cột dữ liệu trong bảng
    - Mỗi dòng trong bảng khách hàng lưu trữ thông tin về một khách hàng trong thực tế

MKH	TEN_KH	PHAI	DIA_CHI	DT	EMAIL
KH001	Trần Văn An	0	123 Nguyễn Du	8123456	tvan@yahoo.com
KH002	Nguyễn Thanh An	0	30 Lê Thánh Tôn	9852147	ntan@yahoo.com
KH003	Lê Thanh Thảo	1	22bis Pasteur	8976431	ltthao@gmail.com

Quan hệ

- Là thành phần được dùng để tạo mối liên kết giữa các bảng dữ liệu với nhau nhằm đảm bảo tính nhất quán, đúng đắn của dữ liệu trong CSDL
- Có ba loại quan hệ chính:
  - Quan hệ 1 – 1
  - Quan hệ 1 – nhiều
  - Quan hệ nhiều – nhiều

Quan hệ

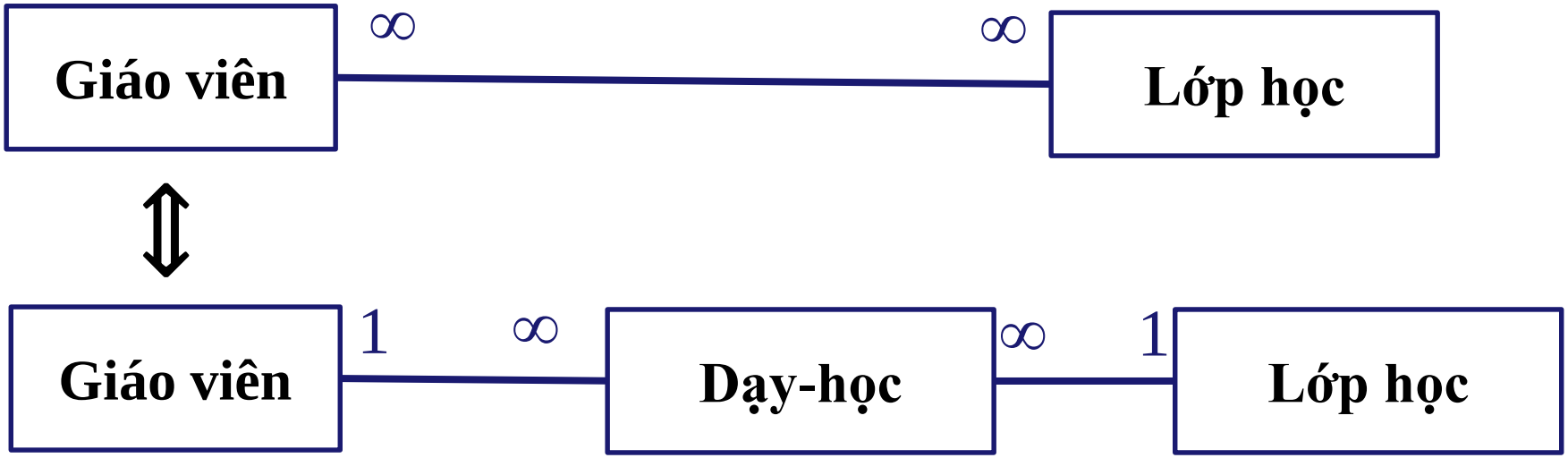
- Quan hệ 1 – 1 (One to One): Mô tả mối quan hệ giữa hai bảng mà trong đó một dòng dữ liệu bên bảng này có liên hệ với duy nhất với một dòng dữ liệu bên bảng kia và ngược lại

Quan hệ

- Quan hệ 1 – nhiều (One to Many): Mô tả mối quan hệ giữa hai bảng mà trong đó một dòng dữ liệu bên bảng này có liên hệ với nhiều dòng dữ liệu bên bảng kia và một dòng dữ liệu bên bảng kia sẽ có liên hệ với duy nhất với một dòng dữ liệu bên bảng này. Quan hệ này thường gặp nhất trong CSDL

Quan hệ

- Quan hệ nhiều – nhiều (Many to Many): Mô tả mối quan hệ giữa hai bảng mà trong đó một dòng dữ liệu bên bảng này có liên hệ với nhiều dòng dữ liệu bên bảng kia và ngược lại. Trong CSDL không lưu trữ quan hệ nhiều nhiều vì vậy khi gặp quan hệ này, chúng ta sẽ chuyển thành các quan hệ một nhiều



SQL

- SQL (Structured Query Language) là ngôn ngữ truy vấn dữ liệu
- Là loại ngôn ngữ cho phép thực hiện các thao tác rút trích, tính toán, cập nhật trên các dữ liệu được lưu trữ trong CSDL

CSDL MySQL

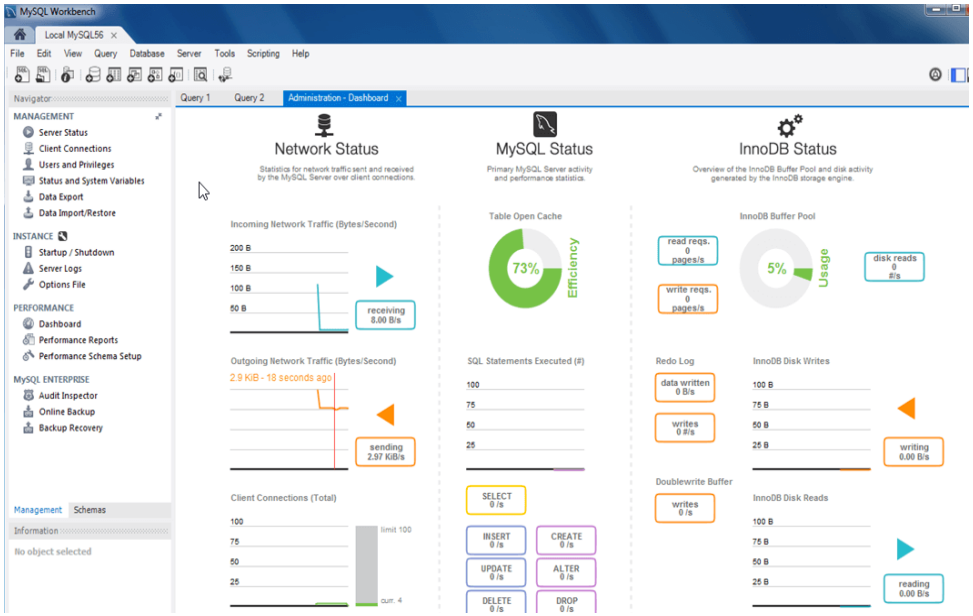
- Giới thiệu
- Đặc điểm
- Các tập tin vật lý lưu trữ CSDL
- Quy tắc đặt tên
- Các thao tác

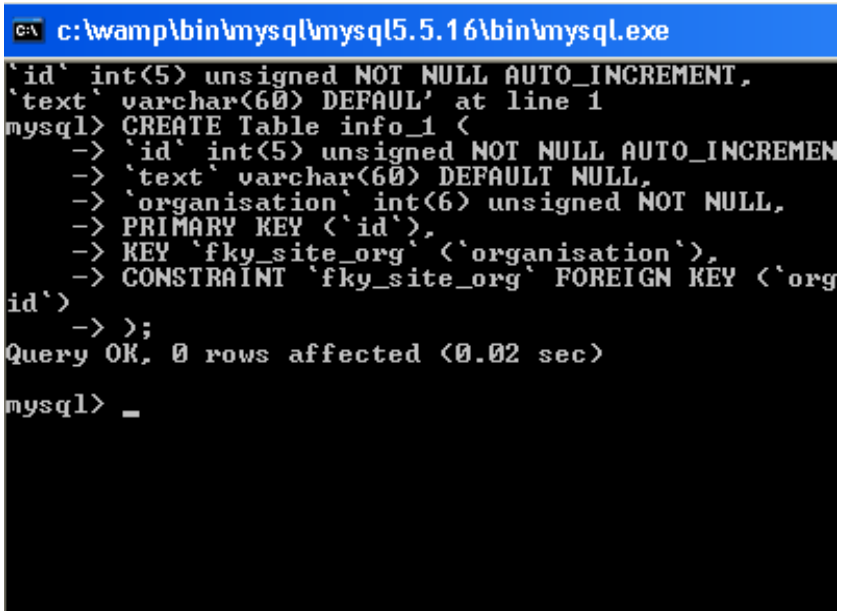
Giới thiệu SQL

- CSDL **MySQL** là tập hợp các đối tượng: bảng, bảng ảo,… cho phép người dùng lưu trữ và truy xuất các thông tin đã được tổ chức và lưu trữ bên trong đó.

Đặc điểm SQL

- Sử dụng cho các ứng dụng Web có quy mô vừa và nhỏ.
- Để thực hiện các thao tác trên CSDL, có thể sử dụng giao diện đồ họa hay dùng dòng lệnh (command line)





Đặc điểm SQL

- Các tập tin vật lý lưu trữ CSDL
  - Mỗi bảng sẽ được lưu trữ dưới ba tập tin vật lý:
    - .frm : lưu định dạng (cấu trúc) của bảng
    - .MYD : lưu nội dung của bảng
    - .MYI : lưu chỉ mục của bảng
- Các tập tin này sẽ được tự động lưu trữ trong thư mục: wamp\mysql\data\name_database

Đặc điểm SQL

❖ Chiều dài của tên CSDL, bảng, chỉ mục, cột, định danh

Loại	Chiều dài tối đa (bytes)	Chiều dài tối đa (ký tự không dấu)
CSDL (database)	64	64
Bảng (Table)	64	64
Chỉ mục (Index)	64	64
Cột (Column)	64	64
Định danh (Alias)	255	255

Đặc điểm SQL

- Quy tắc đặt tên:
  - Tên không kết thúc bằng khoảng trắng
  - Tên CSDL không có các ký tự /, \, ., :, *, ‘’, <, >
  - Tên bảng không có các ký tự /, \, ., :, *, ‘’, <, >, |
  - Chiều dài của tên tối đa là 64 ký tự không dấu. Khi sử dụng các ký tự đa byte thì chiều dài sẽ dựa trên tổng số byte của tất cả các ký tự được dùng.

Thao tác MySQL

- Tạo CSDL
  - Cú pháp:

CREATE DATABASE [IF NOT EXISTS] ten_CSDL;

```
mysql> create database ctk42;
Query OK, 1 row affected (0.01 sec)

mysql>
```

Thao tác MySQL

- Hiển thị các CSDL
  - Cú pháp:

SHOW DATABASES;

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| ctk42              |
| information_schema |
| mysql              |
| performance_schema |
| sakila             |
| sys                |
| test               |
| test2              |
| world              |
+--------------------+
9 rows in set (0.00 sec)
```

Thao tác MySQL

- Xóa Cơ sở Dữ liệu
  - Cú pháp:

DROP DATABASE [IF EXISTS] database_name;

```
mysql> drop database test;
Query OK, 0 rows affected (0.02 sec)
```

Thao tác MySQL

- Chọn CSDL để làm việc
  - Cú pháp:

USE database_name;

Bảng

- **KHÁI NIỆM**
- **THUỘC TÍNH**
- **THAO TÁC VỚI BẢNG**

Khái niệm

- Dùng để lưu trữ thông tin của những đối tượng, thực thể trong thế giới thực muốn được lưu trữ vào máy tính
- Các thông tin trong bảng sẽ được tổ chức thành các dòng (row) và các cột (column).
- Mỗi dòng thông tin trong bảng là duy nhất do có một hoặc nhiều cột làm khóa chính. Dữ liệu của cột làm khóa chính không trùng lắp trong bảng
- Các bảng thường có quan hệ với nhau giúp trao đổi và chia sẻ thông tin

Thuộc tính

- **Tên bảng** (table name)
  - Do người dùng tạo ra
  - Duy nhất trong CSDL
- **Tên cột**
  - Do người dùng tạo ra
  - Duy nhất trong bảng
- **Kiểu dữ liệu** (type): Xác định loại dữ liệu được lưu trữ trên từng cột

Thuộc tính

- **Kiểu dữ liệu** (type): Xác định loại dữ liệu được lưu trữ trên từng cột
  - Các kiểu dữ liệu số nguyên

Kiểu dữ liệu	Kích thước	Miền giá trị
TINYINT	1 byte	-127 => 128 hay 0...255
SMALLINT	2 bytes	-32768 => 32767 hay 0...65535
MEDIUMINT	3 bytes	-8388608 => 838860 hay 0...16777125
INT/INTEGER	4 bytes	-2^{31} => 2^{31}-1 hay 0...2^{32}-1
BIGINT	8 bytes	-2^{63} => 2^{63}-1 hay 0...2^{64}-1

  - Các kiểu dữ liệu true/false

Kiểu dữ liệu	Kích thước	Miền giá trị
BOOL/BOOLEAN	1 byte	Có hai giá trị là true hoặc false

Thuộc tính

❖ **Kiểu dữ liệu** (type):

- Kiểu dữ liệu dạng số thập phân: decimal và numeric
  - Là những kiểu dữ liệu được dùng để lưu trữ những giá trị số cụ thể.
  - Giá trị được lưu với định dạng nhị phân
  - Cú pháp: **Decimal(M[, N]**

  Trong đó: M là tổng ký số và N là số ký số thập phân

Kiểu dữ liệu	Kích thước	Miền giá trị
Decimal/Numeric	4 byte	Phụ thuộc vào khi định nghĩa cột

- Các kiểu dữ liệu số thực

Kiểu dữ liệu	Kích thước	Miền giá trị
Float	4 bytes	-3.402823466E+38 => -1.175494351E-38; 0; 1.175494351E-38 => 3.402823466E+38
Double	8 bytes	-1.7976931348623157E+308 => -2.2250738585072014E-308; 0; 2.2250738585072014E-308 => 1.7976931348623157E+308

Thuộc tính

- **Kiểu dữ liệu** (type):
  - Các kiểu dữ liệu dạng ngày/giờ (date/time)

Kiểu dữ liệu	Miền giá trị	Diễn giải
Date	'1000-01-01' => '9999-12-31'	Ngày với định dạng yyyy-mm-dd
Datetime	'1000-01-01 00:00:00' => '9999-12-31 23:59:59'	Ngày giờ với định dạng yyyy-mm-dd hh:mm:ss
Time	'00:00:00' => '23:59:59'	Giờ với định dạng hh:mm:ss
Year[(2|4)]	4 ký số: '1901' => '2155' 2 ký số: '1970' => '2069'	Năm với định dạng 2 ký số hoặc 4 ký số
Timestamp [(kích cỡ định dạng)]	'1970-01-01 00:00:01'	Timestamp trình bày dưới dạng yyyy-mm-dd hh:mm:ss

Thuộc tính

- **Kiểu dữ liệu** (type):
  - Các kiểu dữ liệu dạng ngày/giờ (date/time)
    - Bảng kích cỡ định dạng (TIMESTAMP)

Kiểu dữ liệu	Định dạng
Timestamp	yyyymmddhhmmss
Timestamp(14)	yyyymmddhhmmss
Timestamp(12)	yymmddhhmmss
Timestamp(10)	yymmddhhmm
Timestamp(8)	yyyymmdd
Timestamp(6)	yymmdd
Timestamp(4)	yymm
Timestamp(2)	yy

Thuộc tính

❖ **Kiểu dữ liệu** (type):

- Các kiểu dữ liệu kiểu chuỗi

Kiểu dữ liệu	Miền giá trị	Diễn giải
Char	1 => 255 ký tự	Chuỗi cố định
Varchar	1 => 255 ký tự	Chuỗi động
Tinyblob	1 => 2^8 -1 bytes (255 bytes)	Kiểu đối tượng nhị phân cỡ 255 ký tự
Tinytext	1 => 2^8-1 ký tự (255 ký tự)	Kiểu đối tượng chuỗi kích cỡ 255 ký tự
Blob	1 => 2^{16}-1 bytes (65535 bytes)	Kiểu blob cỡ 65535 ký tự
Text	1 => 2^{16}-1 ký tự (65535 ký tự)	Kiểu chuỗi dạng văn bản cỡ 65535 ký tự
Mediumblob	1 => 2^{24}-1 bytes (16777215 bytes)	Kiểu blob vừa cỡ 16777215 ký tự
Mediumtext	1 => 2^{24}-1 ký tự (16777215 ký tự)	Kiểu chuỗi dạng văn bản vừa 16777215 ký tự
Longblob	1 => 2^{32}-1 bytes (4GB)	Kiểu blob lớn khoảng 4GB ký tự
Longtext	1 => 2^{32}-1 ký tự (4GB)	Kiểu chuỗi dạng văn bản lớn khoảng 4GB ký tự

Thuộc tính

- **Kiểu dữ liệu** (type):
  - Các kiểu dữ liệu kiểu chuỗi
    - So sánh khác nhau giữa kiểu char và kiểu varchar

Giá trị	Char(4)	Số bytes	Varchar(4)	Số bytes
''	' '	4 bytes	''	1 byte
'ab'	'ab '	4 bytes	'ab'	3 bytes
'abcd'	'abcd'	4 bytes	'abcd'	5 bytes
'abcdefgh'	'abcd'	4 bytes	'abcd'	5 bytes

Thuộc tính

- Độ dài dữ liệu (length/value): Quy định độ dài dữ liệu mà cột sẽ lưu trữ đối với kiểu dữ liệu chuỗi hoặc số
- Kiểu hiển thị (collation): Quy định bảng mã hiển thị cho dữ liệu trong cột.
- Thuộc tính (attribute): Quy định thuộc tính cho cột, mặc định là không quy định
- Cho phép để trống dữ liệu (NULL): Quy định cột có thể để trống hay không khi thêm, cập nhật dữ liệu
- Giá trị mặc định (default):Là giá trị sẽ thêm vào cho cột khi thêm mới mẩu tin mà người dùng không nhập giá trị cho cột
- Thuộc tính mở rộng (extra): Cho phép thiết lập thuộc tính auto increment (cột có giá trị tự động tăng dần khi thêm mới mẩu tin) cho khóa chính
- Ghi chú (comment): Chuỗi chú thích cho cột

Thao tác với bảng

- Tạo bảng
- Thay đổi cấu trúc bảng
  - Thêm cột mới trong bảng
  - Sửa đổi kiểu dữ liệu của cột
  - Hủy cột trong bảng
- Xóa bảng

Tạo bảng

- Tạo bảng đơn giản
  - **Cú pháp**:

    CREATE TABLE [**IF NOT EXISTS**] table_name(
    <column name><type> [<default value>] [column constraints],
    ...
    <column name><type> [<default value>] [column constraints],
    <table constraint>,
    ...
    <table constraint>
) type=table_type

Tạo bảng

- Tạo bảng đơn giản
  - **Cú pháp**:

    CREATE TABLE [IF NOT EXISTS] table_name(
    <column name><type> [<**default value**>] [column constraints],
    ...
    <column name><type> [<**default value**>] [column constraints],
    <table constraint>,
    ...
    <table constraint>
) type=table_type

Tạo bảng

- Tạo bảng đơn giản
  - **Cú pháp**:

    CREATE TABLE [IF NOT EXISTS] table_name(
    <column name><type> [<default value>] [**column constraints**],
    ...
    <column name><type> [<default value>] [**column constraints**],
    <**table constraint**>,
    ...
    <**table constraint**>
) type=table_type

Ràng buộc

- **NOT NULL**: Ràng buộc này yêu cầu giá trị của cột không được phép là NULL
- **PRIMARY KEY** (ràng buộc khóa chính): Ràng buộc này định nghĩa một cột hoặc một tổ hợp các cột xác định duy nhất mỗi dòng trong bảng
- **UNIQUE**: ràng buộc yêu cầu các giá trị của cột là phân biệt. Chú ý với ràng buộc này giá trị của cột có thể là NULL nếu ràng buộc NOT NULL không được áp dụng trên cột
- Phân biệt giữa **primary** và **unique**

Primary key

Một bảng chỉ có một khóa chính, có thể có một hay nhiều cột tham gia làm khóa chính => Primary key chỉ xuất hiện một lần khi tạo cấu trúc bảng

Unique

Unique kiểm tra tính duy nhất của dữ liệu trên các cột trong bảng => Unique có thể được xuất hiện nhiều lần khi tạo cấu trúc bảng

Ràng buộc

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

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

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

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

Ràng buộc

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

```
CREATE TABLE employees (
employeeNumber int(11) NOT NULL PRIMARY KEY ,
lastName varchar(50) NOT NULL,
firstName varchar(50) NOT NULL,
extension varchar(10) NOT NULL,
email varchar(100) NOT NULL,
officeCode varchar(10) NOT NULL,
reportsTo int(11) default NULL,
jobTitle varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Ràng buộc

- Ví dụ: Tạo bảng employees với khóa chính xác định theo ràng buộc mức bảng

```
CREATE TABLE employees (
employeeNumber int(11) NOT NULL,
lastName varchar(50) NOT NULL,
firstName varchar(50) NOT NULL,
extension varchar(10) NOT NULL,
email varchar(100) NOT NULL,
officeCode varchar(10) NOT NULL,
reportsTo int(11) default NULL,
jobTitle varchar(50) NOT NULL,
PRIMARY KEY (employeeNumber)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Ràng buộc

- Đặt tên ràng buộc: **CONSTRAINT <name> <constraint>**
- **Mục đích:** khi cập nhật dữ liệu vi phạm ràng buộc, hệ quản trị CSDL thường đưa tên ràng buộc vào thông báo lỗi. Hơn nữa có thể sử dụng tên ràng buộc khi sửa đổi hoặc xóa ràng buộc

```
CREATE TABLE employees (
employeeNumber int(11) NOT NULL CONSTRAINT
emp_id_pk PRIMARY KEY,
lastName varchar(50) NOT NULL,
firstName varchar(50) NOT NULL,
extension varchar(10) NOT NULL,
email varchar(100) NOT NULL,
officeCode varchar(10) NOT NULL,
reportsTo int(11) default NULL,
jobTitle varchar(50) NOT NULL,
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Ràng buộc

❖ FOREIGN KEY (Ràng buộc khóa ngoài):

```
CREATE TABLE city (
city_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
city VARCHAR(50) NOT NULL,
country_id SMALLINT UNSIGNED NOT NULL,
last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON
UPDATE CURRENT_TIMESTAMP,
PRIMARY KEY(city_id),
CONSTRAINT fk_city_country FOREIGN KEY (country_id)
REFERENCES country (country_id) ON DELETE RESTRICT ON
UPDATE CASCADE
)
```

Ràng buộc

- Tạo cột duy nhất: `UNIQUE (các_cột_duy_nhất)`
- Ví dụ1: tạo bảng *hang_sua* như trên với cột *Ten_hang_sua* là duy nhất

```
CREATE TABLE hang_sua (
Ma_hang_sua varchar(20) NOT NULL PRIMARY KEY,
Ten_hang_sua varchar(100) NOT NULL UNIQUE,
Dia_chi varchar(200),
Dien_thoai varchar(20),
Email varchar(100)
)
```

- Ví dụ 2: tạo bảng hang_sua như trên với cột Ten_hang_sua và email là duy nhất

```
CREATE TABLE hang_sua (
Ma_hang_sua varchar(20) NOT NULL PRIMARY KEY,
Ten_hang_sua varchar(100) NOT NULL,
Dia_chi varchar(200),
Dien_thoai varchar(20),
Email varchar(100),
UNIQUE(Ten_hang_sua, Email)
)
```

Ràng buộc

- Xác định cột tự tăng giá trị **Auto_Increment**
  - Tạo ra cột có giá trị tự động tăng dần
  - Thuộc tính auto_increment chỉ có thể thiết lập cho cột có kiểu dữ liệu là kiểu số nguyên.
  - Trong một bảng, chỉ có thể có một cột có thuộc tính auto_increment, cột này phải là khóa và không thiết lập giá trị mặc định DEFAULT

```
CREATE TABLE loai
(
      Ma_loai int NOT NULL AUTO_INCREMENT
PRIMARY KEY,
      Ten_loai VARCHAR(30) NOT NULL
)
```

Tạo bảng

- Tạo bảng đơn giản
  - **Cú pháp**:

```
CREATE TABLE [IF NOT EXISTS] table_name(
<column name><type> [<default value>] [column
constraints],
...
<column name><type> [<default value>] [column
constraints],
<table constraint>,
...
<table constraint>
) type=table_type
```

Thay đổi cấu trúc bảng

- Thêm, xóa, sửa các cột của bảng
- Thêm và xóa các ràng buộc
- Cú pháp của lệnh **ALTER TABLE** :

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

Các tùy chọn:

```
ADD [COLUMN] <column_definition>
MODIFY [COLUMN] <create_definition>
CHANGE [O_COLUMN] [N_COLUMN] <create_definition>
DROP [COLUMN] <column_name>
ADD <table_constraint>
DROP <constraint_name>
```

Thay đổi cấu trúc bảng

❖ Ví dụ thêm cột

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

❖ Ví dụ sửa kiểu dữ liệu của cột

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

```
ALTER TABLE employees
CHANGE detail detail text;
```

❖ Ví dụ xóa cột

```
ALTER TABLE employees
DROP officeCode
```

Xóa bảng

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

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

Phát biểu SQL

- Truy vấn đơn giản
- Sử dụng mệnh đề UNION trong truy vấn
- Các hàm xử lý trong MySQL
- Truy vấn nhóm
- Truy xuất dữ liệu từ nhiều bảng
- Truy vấn con (Subquery)
- Thêm, sửa, xóa dữ liệu trong bảng

Truy vấn đơn giản SELECT … FROM

- Chọn ra dữ liệu của các cột có trong một bảng
- Cú pháp:

```
SELECT tên danh sách các cột
FROM tên bảng
[WHERE điều kiện chọn]
[GROUP BY nhóm]
[HAVING điều kiện chọn nhóm]
[ORDER BY Tên cột sắp xếp [DESC, ASC]]
[LIMIT giới hạn số lượng mẩu tin];
```

```
SELECT * FROM employees
```

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

Truy vấn đơn giản SELECT … FROM

- Từ khóa **DISTINCT** sẽ loại bỏ dữ liệu trùng lặp từ câu lệnh SELECT
- Cú pháp:

```
SELECT DISTINCT tên danh sách các cột
FROM tên bảng
```

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

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

Truy vấn đơn giản SELECT … FROM

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

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

orderNumber	lineTotal
10100	4080
10100	2754.5
10100	1660.12
10100	1729.21
10101	2701.5
10101	4343.56
10101	1463.8500000000001

Mệnh đề WHERE

- Mệnh đề **WHERE** của câu lệnh SELECT cho phép tìm kiếm theo điều kiện hoặc tiêu chí tìm kiếm là một điều kiện nhất định
- Cú pháp:

```
SELECT tên danh sách các cột
FROM tên bảng
[WHERE điều kiện chọn]
```

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

Mệnh đề WHERE

- Các phép toán thường gặp trong điều kiện chọn
  - **So sánh**: >, >=, <, <=, =, !=, <>
  - **Logic**: and, or, not, in, not in, between, like, not like

Mệnh đề WHERE

❖ Cú pháp:

SELECT tên danh sách các cột

FROM tên bảng

[WHERE điều kiện chọn 1 AND/OR điều kiện chọn 2 …]

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

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

Mệnh đề WHERE

❖ Cú pháp:

SELECT danh sách các cột

FROM tên bảng

WHERE cột IN ("giá trị 1","giá trị 2"…)

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

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

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

Mệnh đề WHERE

❖ Cú pháp:

```
SELECT danh sách các cột
FROM tên bảng
WHERE cột NOT IN ("giá trị 1","giá trị 2"…)
```

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

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

Mệnh đề WHERE

❖ Cú pháp:

```
SELECT danh sách các cột
FROM tên bảng
WHERE column_1 BETWEEN lower_range AND upper_range
```

```
SELECT danh sách các cột
FROM tên bảng
WHERE column_1 >= lower_range AND column_1 <= upper_range
```

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

productCode	ProductName	buyPrice
S10_1949	1952 Alpine Renault 1300	98.58
S24_3856	1956 Porsche 356A Coupe	98.3
S12_1108	2001 Ferrari Enzo	95.59
S12_1099	1968 Ford Mustang	95.34
S18_1984	1995 Honda Civic	93.89

Mệnh đề WHERE

❖ Cú pháp:

```
SELECT danh sách các cột
FROM tên bảng
WHERE column_1 NOT BETWEEN lower_range AND upper_range
```

```
SELECT danh sách các cột
FROM tên bảng
WHERE column_1 <= lower_range OR column_1 >= upper_range
```

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

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

Mệnh đề WHERE

- LIKE cho phép thực hiện việc tìm kiếm thông tin dựa trên sự so sánh ký tự (‘*giống như*’)
- MySQL cung cấp cho hai ký tự đại diện sử dụng với LIKE, đó là **%** và **_**
  - Ký tự đại diện tỷ lệ phần trăm (%) đại diện cho bất kỳ chuỗi có thể không có hoặc có nhiều ký tự
  - Gạch dưới (_) chỉ đại diện cho một ký tự duy nhất.
- Cú pháp:

```
SELECT danh sách các cột
FROM tên bảng
WHERE column LIKE ‘%xxx%’
```

```
SELECT danh sách các cột
FROM tên bảng
WHERE column LIKE ‘_xxx_’
```

Mệnh đề WHERE

- VD1: Tìm kiếm những nhân viên có tên bắt đầu với ký tự 'a'

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

employeeNumber	lastName	firstName
1143	Bow	Anthony
1611	Fixter	Andy

- VD2: Tìm kiếm tất cả các nhân viên có họ kết thúc với chuỗi 'on'

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

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

- VD3: Tìm tất cả các nhân viên mà họ của các nhân viên này có chứa cụm 'on'

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

employeeNumber	lastName	firstName
1056	Patterson	Mary
1088	Patterson	William
1102	Bondur	Gerard

Mệnh đề WHERE

- Dùng NOT kèm với LIKE để hàm chứa ý nghĩa phủ định
- Cú pháp:

```
SELECT danh sách các cột
FROM tên bảng
WHERE column NOT LIKE ‘%xxx%’
```

```
SELECT danh sách các cột
FROM tên bảng
WHERE column NOT LIKE ‘_xxx_’
```

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

employeeNumber	lastName	firstName
1002	Murphy	Diane
1056	Patterson	Mary
1076	Firrelli	Jeff
1088	Patterson	William
1165	Jennings	Leslie
1166	Thompson	Leslie
1188	Firrelli	Julie

Mệnh đề WHERE

- **Chú ý**: Trường hợp chuỗi tìm kiếm của lại bắt đầu bởi một ký tự đại diện, Mysql cung cấp cho ký tự '\' để chỉ ra rằng các ký tự đại diện đi sau đó được sử dụng theo đúng nghĩa đen chứ không còn là ý nghĩa là ký tự đại diện nữa.
- **Ví dụ**: Tìm các sản phẩm mà mã của chúng có chứa chuỗi '**_20**'

```
SELECT productCode, productName
FROM products
WHERE productCode LIKE '%\_20%'
```

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

Mệnh đề WHERE

- Phép toán **IS NULL**
- Cú pháp:

```
SELECT tên danh sách các cột
FROM tên bảng
[WHERE điều kiện chọn IS NULL]
```

: *tìm các giá trị không xác định*

```
SELECT customerName,
salesRepEmployeeNumber
FROM customers
WHERE salesRepEmployeeNumber is NULL
```

customerName	salesRepEmployeeNumber
Havel & Zbyszek Co	NULL
Porto Imports Co.	NULL
Asian Shopping Network, Co	NULL
Natürlich Autos	NULL
ANG Resellers	NULL
Messner Shopping Network	NULL

Mệnh đề WHERE

- Có thể sử dụng mệnh đề WHERE để liên kết dữ liệu của nhiều bảng trong truy vấn
- Cú pháp:

```
SELECT tên danh sách các cột
FROM Tên_bảng_1, Tên_bảng_2, ...
WHERE Tên_bảng_1.tên_cột = Tên_bảng_2.tên_cột
[ORDER BY Tên_cột_sắp_xếp [DESC, …]]
```

```
SELECT Ma_sua, Ten_sua,Ten_hang_sua
FROM HangSua, Sua
WHERE HangSua.Ma_hang_sua = Sua.Ma_hang_sua
AND
Sua.Ma_hang_sua NOT IN("DL", "DS", "VNM")
```

Ma_sua	Ten_sua	Ten_hang_sua
AB0001	Gain Advance	Abbott
AB0002	Gain IQ	Abbott
AB0003	Abbott Grow	Abbott
AB0004	Abbott Grow School	Abbott
AB0005	Abbott Pedia Sure	Abbott
AB0006	Similac Neo Sure	Abbott
MJ0001	Enfa Mama A+	Mead Jonhson
MJ0002	EnfaLac	Mead Jonhson
MJ0003	EnfaGrow	Mead Jonhson

Truy vấn có sắp xếp dữ liệu

- Giúp lấy dữ liệu của các cột bên trong bảng đồng thời sắp xếp lại dữ liệu theo thứ tự tăng dần hoặc giảm dần
- Cú pháp:

```
SELECT tên danh sách các cột
FROM tên bảng
[WHERE điều kiện chọn]
[ORDER BY Tên cột sắp xếp [DESC, ASC]]
```

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

FirstName	LastName	jobtitle
Andy	Fixter	Sales Rep
Anthony	Bow	Sales Manager (NA)
Barry	Jones	Sales Rep
Diane	Murphy	President
Foon Yue	Tseng	Sales Rep
George	Vanauf	Sales Rep
Gerard	Hernandez	Sales Rep

Giới hạn số lượng kết quả với LIMIT

- LIMIT cho phép hạn chế các bản ghi trả lại với câu lệnh SELECT
- Cú pháp:

```
SELECT * FROM table_name
LIMIT S, N //lấy ra một số lượng N bản ghi nhất định tính từ một vị trí S nào đó
```

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

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

Sử dụng mệnh đề UNION trong truy vấn

- **UNION** cho phép kết nối dữ liệu của các câu lệnh truy vấn lại với nhau.
- Cú pháp:

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

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

Sử dụng mệnh đề UNION trong truy vấn

- Ghi chú:
  - Mặc định UNION loại bỏ tất cả các hàng trùng lặp từ kết quả ngay cả khi không sử dụng từ khoá DISTINCT
  - Muốn các hàng trùng lặp vẫn còn trong tập hợp kết quả thì sử dụng từ khóa UNION ALL.
  - Khi sử dụng ORDER BY để sắp xếp kết quả với UNION, phải đặt nó ở vị trí cuối cùng trong mệnh đề SELECT

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

customerNumber	contactLastname
249	Accorti
481	Altagar,G M
307	Andersen
1611	Andy
1143	Anthony

Sử dụng mệnh đề UNION trong truy vấn

- Ghi chú:
  - MySQL cũng cung cấp một lựa chọn khác để sắp xếp các kết quả thiết lập dựa trên vị trí cột trong mệnh đề ORDER BY.

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

customerNumber	contactLastname
249	Accorti
481	Altagar,G M
307	Andersen
1611	Andy

Sử dụng mệnh đề UNION trong truy vấn

- Ghi chú:
  - Nếu tên cột không giống nhau trong hai mệnh đề SELECT của phép UNION, MySQL sẽ sử dụng các tên cột của câu lệnh SELECT đầu tiên làm tên cột đầu ra.

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

customerNumber	contactLastname
249	Accorti
481	Altagar,G M
307	Andersen
1611	Andy
1143	Anthony

Các hàm xử lý trong MySQL

- **Các hàm cấu trúc điều khiển**
- **Các hàm chuyển đổi kiểu dữ liệu**
- **Các hàm xử lý chuỗi**
- **Các hàm xử lý số**
- **Các hàm xử lý thời gian**
- **Hàm LAST_INSERT_ID**

Các hàm cấu trúc điều khiển

- Hàm **IF**:là một hàm điều khiển, trả về kết quả là một chuỗi hoặc số dựa trên một điều kiện cho trước.
- Cú pháp:

IF (biểu_thức_so_sánh, biểu_thức_1, biểu_thức_2)

  - Tham số đầu tiên là biểu thức so sánh sẽ được kiểm tra là đúng hay sai. Biểu thức so sánh thì không bằng 0 và không bằng NULL.
  - Khi biểu thức so sánh đúng thì kết quả trả về là biểu thức 1, ngược lại thì kết quả trả về là biểu thức 2

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

customerNumber	customerName	state	country
103	Atelier graphique	N/A	France
112	Signal Gift Stores	NV	USA
114	Australian Collectors, Co.	Victoria	Australia
119	La Rochelle Gifts	N/A	France
121	Baane Mini Imports	N/A	Norway
124	Mini Gifts Distributors Ltd.	CA	USA

Các hàm cấu trúc điều khiển

❖ Hàm **IF** với chức năng tổng hợp.

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

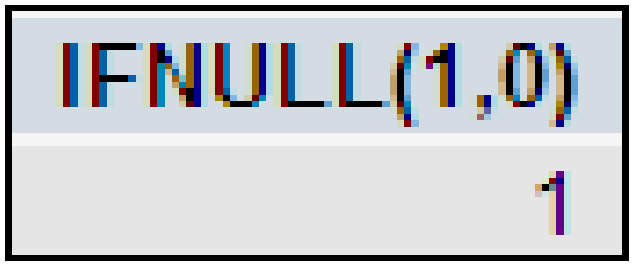
Shipped	Cancelled
303	6

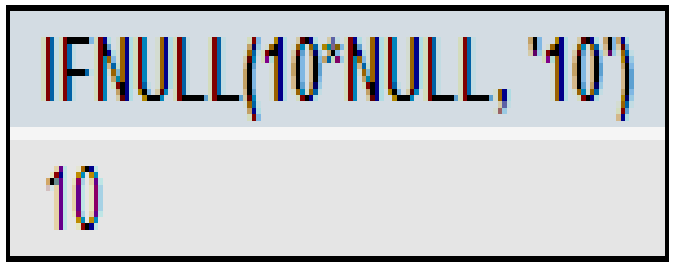
Các hàm cấu trúc điều khiển

- Hàm **IFNULL**: kiểm tra giá trị NULL
- Cú pháp:

IFNULL(biểu thức 1, biểu thức 2)

- Nếu biểu thức 1 khác NULL thì hàm IFNULL có kết quả trả về là biểu thức 1, ngược lại thì kết quả trả về là biểu thức 2
- Hàm IFNULL trả về giá trị số hoặc chuỗi tùy thuộc vào nội dung trong các biểu thức.





Các hàm cấu trúc điều khiển

- Hàm **NULLIF**: So sánh sự khác biệt
- Cú pháp:

`NULLIF(biểu thức 1, biểu thức 2)`

  - Nếu biểu thức 1 bằng biểu thức 2 thì hàm NULLIF có kết quả trả về là NULL, ngược lại thì kết quả trả về là biểu thức 1.

NULLIF(1,1)
NULL

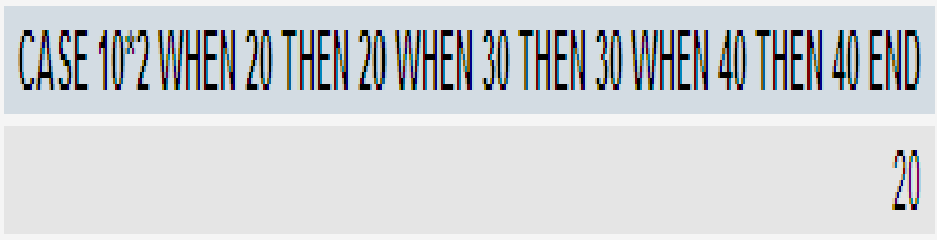
NULLIF(1,2)
1

Các hàm cấu trúc điều khiển

- Hàm **CASE**: thực hiện việc so sánh một giá trị hay một biểu thức với hàng loạt các giá trị khác để đưa về một kết quả thích hợp với giá trị hay biểu thức đã đem so sánh.
- CASE dạng đơn giản:
  - Cú pháp:

```
CASE biểu_thức_giá_trị
WHEN giá_trị_so_sánh THEN kết_quả
[WHEN giá_trị_so_sánh THEN kết_quả ...]
[ELSE kết_quả]
END
```

```
SELECT CASE 10*2
WHEN 20 THEN 20
WHEN 30 THEN 30
WHEN 40 THEN 40
END
```



Các hàm cấu trúc điều khiển

- CASE dạng có điều kiện:
  - Cú pháp:

```
CASE
WHEN điều_kiện_1 THEN kết_quả_1
WHEN điều_kiện_2 THEN kết_quả_2
...
[WHEN điều_kiện_n-1 THEN kết_quả_n-1]
[ELSE kết_quả_n]
END
```

```
SELECT CASE
WHEN 10*2=30 THEN ‘30’
WHEN 10*2=40 THEN ‘40’
ELSE ‘10*2=20’
END
```

```
CASE WHEN 10*2=30 THEN '30' WHEN 10*2=40 THEN '40' ELSE '10*2=20' END
```

```
10*2=20
```

Các hàm chuyển đổi kiểu dữ liệu

- Hàm **CAST**
  - Dùng để chuyển đổi một giá trị hoặc biểu thức sang một kiểu dữ liệu khác.
  - Kết quả trả về là giá trị hoặc biểu thức với kiểu dữ liệu mới
- Cú pháp:

```
CAST(biểu_thức AS kiểu_dữ_liệu)
```

Kiểu dữ liệu có thể là một trong các kiểu sau: BINARY[(N)], CHAR[(N)], DATE, DATETIME, DECIMAL, SIGNED [INTEGER], TIME, UNSIGNED [INTEGER]

```
CAST(20071212 AS DATE)
2007-12-12
```

Các hàm chuyển đổi kiểu dữ liệu

- Hàm **CONVERT**
  - **Dạng 1**: dùng để chuyển đổi một giá trị hoặc biểu thức sang một kiểu dữ liệu khác.
  - Kết quả trả về là giá trị hoặc biểu thức với kiểu dữ liệu mới
  - Cú pháp:

```
CONVERT(biểu_thức, kiểu_dữ_liệu)
```

```
SELECT CONVERT(20071212, DATE)
```

CONVERT(20071212, DATE)
2007-12-12

Các hàm chuyển đổi kiểu dữ liệu

- Hàm **CONVERT**
  - **Dạng 2**: dùng để chuyển một giá trị hoặc một biểu thức sang một kiểu hiển thị khác.
  - Kết quả trả về là giá trị hoặc biểu thức dưới dạng hiển thị mới
  - Kiểu hiển thị có thể là một trong các kiểu sau: utf8, latin1..,latin7, ascii, binary...
  - Cú pháp:

```
CONVERT(biểu_thức USING kiểu_hiển_thị)
```

```
SELECT CONVERT('cats and dogs' USING latin2
```

CONVERT("cats and dogs" USING latin2)
cats and dogs

Các hàm xử lý chuỗi

- Hàm **CHAR_LENGHT()**, **CHARACTER_LENGTH()** và **LENGTH()**
  - Kết quả trả về là chiều dài của chuỗi (str) nhưng theo hai dạng là chiều dài tính theo ký tự (**char_length()**, **character_length()**) và chiều dài tính theo byte (**length**)
  - Cú pháp:

```
CHAR_LENGTH(str)
CHARACTER_LENGTH(str)
```

  - Kết quả trả về số ký tự có trong chuỗi bao gồm cả khoảng trắng

```
SELECT CHAR_LENGTH("anh và em")
```

CHAR_LENGTH("anh và em")
9

Các hàm xử lý chuỗi

- Hàm **CHAR_LENGHT()**, **CHARACTER_LENGTH()** và **LENGTH()**
  - Cú pháp:

`LENGTH(str)`

  - Kết quả trả về chiều dài của chuỗi được tính bằng byte

```
SELECT LENGTH(“anh và em”)
```

LENGTH("anh và em")
10

Các hàm xử lý chuỗi

- Hàm **CONCAT()**
  - Nối hai hoặc nhiều chuỗi thành một chuỗi mới.
  - Nếu các đối số là số, chúng sẽ được chuyển đổi thành chuỗi trước khi nối.
  - Nếu bất kỳ đối số trong danh sách đối số là NULL, hàm **CONCAT** sẽ trả về NULL
  - Cú pháp:

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

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

fullname
Schmitt, Carine
King, Jean
Ferguson, Peter
Labrune, Janine
Bergulfsen, Jonas

Các hàm xử lý chuỗi

- Hàm **CONCAT_WS ( )**
  - Nối hai hay nhiều hơn hai chuỗi với một dấu phân cách được xác định trước.
  - Cú pháp:

```
CONCAT_WS(chỉ định cách,str1,str2,...)
```

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

fullname
Schmitt; Carine
King; Jean
Ferguson; Peter
Labrune; Janine
Bergulfsen; Jonas

Các hàm xử lý chuỗi

- Hàm **CONCAT_WS ( )**
  - **Bài tập:** Sử dụng hàm **CONCAT_WS( )** để tạo ra cột địa chỉ của khách hàng từ bảng **customers** như hình sau

Customer_Address
Schmitt Carine 54, rue Royale44000 NantesFrance
King Jean8489 Strong St.83030 Las VegasUSA
Ferguson Peter636 St Kilda RoadLevel 33004 MelbourneAustralia
Labrune Janine 67, rue des Cinquante Otages44000 NantesFrance
Bergulfsen Jonas Erling Skakkes gate 784110 StavernNorway
Nelson Susan5677 Strong St.97562 San RafaelUSA
Piestrzeniewicz Zbyszek ul. Filtrowa 6801-012 WarszawaPoland
Keitel RolandLyonerstr. 3460528 FrankfurtGermany
Murphy Julie5557 North Pendale Street94217 San FranciscoUSA
Lee Kwai897 Long Airport Avenue10022 NYCUSA

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

Các hàm xử lý chuỗi

- Hàm **LOWER()**
  - Kết quả trả về là một chuỗi sau khi đã chuyển các ký tự trong chuỗi thành chữ thường
  - Cú pháp:

```
LOWER(str)
```

- Hàm **UPPER()**
  - Kết quả trả về là một chuỗi sau khi đã chuyển các ký tự trong chuỗi thành chữ hoa
  - Cú pháp:

```
UPPER(str)
```

Các hàm xử lý chuỗi

❖ Hàm **LOWER()** và hàm **UPPER()**

```
SELECT UPPER(Ten_hang_sua) as 'Tên viết hoa',
LOWER(Email) as 'Email viết thường'
FROM HangSua
```

Tên viết hoa	Email viết thường
VINAMILK	vinamilk@vnm.com
NUTIFOOD	nutifood@ntf.com
ABBOTT	abbott@ab.com
DAISY	daisy@ds.com
DUTCH LADY	dutchlady@dl.com
DUMEX	dumex@dm.com
MEAD JONHSON	meadjohn@mj.com

Các hàm xử lý chuỗi

- Hàm **LEFT()** và **RIGHT()**
  - Kết quả trả về là một chuỗi con được trích ra từ chuỗi gốc. Trong đó chuỗi con được trích ra có thể bắt đầu từ bên trái (LEFT()) hay bên phải (RIGHT()) của chuỗi.
  - Cú pháp

```
LEFT(str, số_byte)
RIGHT(str, số_byte)
```

```
SELECT LEFT("anh và em", 3)
```

left("anh và em", 3)
anh

```
SELECT RIGHT("anh và em", 2)
```

RIGHT('anh và em', 2)
em

Các hàm xử lý chuỗi

- Hàm **MID()** và **SUBSTRING()**
  - Hàm Substring cho phép trích xuất một chuỗi con từ một chuỗi khác, bắt đầu tại vị trí cụ thể và với một độ dài nhất định
  - Cú pháp 1:

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

  - Cú pháp 2:

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

Các hàm xử lý chuỗi

❖ Hàm **MID()** và **SUBSTRING()**

```
SELECT MID('MySQL Workbench', 7);
```

Workbench

```
SELECT substring('MySQL Workbench',7);
```

Workbench

```
SELECT substring('MySQL Workbench' FROM 7);
```

Workbench

```
SELECT substring('MySQL Workbench',7,4);
```

Work

```
SELECT substring('MySQL Workbench', FROM 7 FOR 3);
```

Work

Các hàm xử lý chuỗi

- Hàm **MID()** và **SUBSTRING()**
  - **Chú ý**: Nếu sử dụng giá trị âm cho tham số **pos**, sự bắt đầu của chuỗi con được tính từ cuối của chuỗi.

```
SELECT substr('MySQL Workbench', -8);
```

Workbench

Các hàm xử lý chuỗi

❖ Hàm **REPEAT()**

- Được dùng để lặp lại nhiều lần một chuỗi.
- Kết quả trả về là một chuỗi mới được tạo ra từ chuỗi được lặp lại
- Cú pháp:

```
REPEAT(str, số_lần_lặp)
```

```
SELECT REPEAT("Tôi đi học", 3)
```

REPEAT("Tôi đi học", 3)
Tôi đi họcTôi đi họcTôi đi học

Các hàm xử lý chuỗi

- Hàm **REVERSE()**
  - Kết quả trả về là một chuỗi đảo ngược.
  - Cú pháp:

`REVERSE(str)`

```
SELECT REVERSE('anh va em');
```

```
REVERSE("anh và em")
me àv hna
```

Các hàm xử lý chuỗi

- Hàm REPLACE()
  - Kết quả trả về là một chuỗi mới sau khi tìm và thay thế một chuỗi con trong chuỗi nguồn bằng một chuỗi khác
  - Cú pháp:

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

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

Các hàm xử lý chuỗi

- Hàm **ENCODE():** Dùng để mã hóa một chuỗi
  - Cú pháp:

    `ENCODE(str, khóa)`

    - **str**: là chuỗi sẽ được mã hóa dưới dạng chuỗi nhị phân.
    - **khóa**: là password do chúng ta đặt ra để không cho phép người khác giải mã

```
SELECT ENCODE(“phuong”, “nd1”)
```

```
ENCODE("phuong", "nd1")
_□9□□s
```

Các hàm xử lý chuỗi

- Hàm DECODE() dùng để giải mã thông tin đã bị mã hóa.
- Cú pháp:

```
DECODE(str, khóa)
```

  - **str**: là chuỗi đã bị mã hóa.
  - **khóa**: là mật khẩu được đặt ra khi tiến hành mã hóa. Không có mật khẩu này thì không thể giải mã

ten_dang_nhap	encode(`mat_khau` , 'nd1')
phuong	□□□□X8
thien	□□□□X8
quy	□□□□X8
thy	□□□□X8

```
SELECT ten_dang_nhap,
DECODE(mat_khau, 'nd1')
```

ten_dang_nhap	Giải mã mật khẩu
phuong	123456
thien	123456
quy	123456
thy	123456

Các hàm xử lý chuỗi

- Hàm **SPACE()**
  - Kết quả trả về là một chuỗi có N khoảng trắng
  - Cú pháp:

SPACE(N)

```
SELECT CONCAT( Ten_hs,
SPACE(10) , Dien_thoai )
FROM hang_sua
```

CONCAT(Ten_hs , SPACE(10) , Dien_thoai)
Vinamilk 8794561
Nutifood 7895632
Abbott 8741258
Daisy 5789321
Dutch Lady 7826451
Dumex 6258943
Mead Jonhson 8741258

Các hàm xử lý chuỗi

- Hàm **STRCMP()**
  - Dùng để so sánh chính xác hai chuỗi
  - Kết quả trả về bằng 0 nếu hai chuỗi giống nhau, trả về -1 nếu chuỗi 1 nhỏ hơn chuỗi 2, trả về 1 nếu chuỗi 1 lớn hơn chuỗi 2
  - Cú pháp:

```
STRCMP(str1, str2)
```

```
SELECT STRCMP('text', 'text2')
```

STRCMP('text', 'text2')
-1

```
SELECT STRCMP('text2', 'text')
```

STRCMP('text2', 'text')
1

```
SELECT STRCMP('text', 'text')
```

STRCMP('text', 'text')
0

Các hàm xử lý số

- Hàm **CEILING() / CEIL()**: dùng để làm tròn số theo cận trên
  - Kết quả trả về là số nguyên nhỏ nhất có giá trị không nhỏ hơn X
  - Cú pháp:

```
CEILING(X)
CEIL(X)
```

```
SELECT CEILING(9.327)
```

CEILING(9.327)
10

Các hàm xử lý số

- Hàm **FLOOR()** dùng để làm tròn số theo cận dưới
  - Kết quả trả về là số nguyên lớn nhất có giá trị không lớn hơn X
  - Cú pháp:

FLOOR(X)

```
SELECT FLOOR(9.327)
```

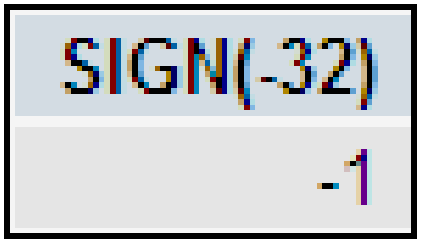
FLOOR(9.327)
9

Các hàm xử lý số

- Hàm **SIGN():** xét dấu của số hay biểu thức'
  - Kết quả trả về là 1 nếu số hay biểu thức là số dương, -1 nếu số âm, 0 nếu số bằng 0
  - Cú pháp:

SIGN(số)

```
SELECT SIGN(-32)
```



Các hàm xử lý thời gian

- Hàm **ADDDATE()** và **DATE_ADD()**: có cùng kết quả trả về là một ngày mới sau khi đã cộng thêm một đơn vị thời gian
  - Cú pháp 1:

```
ADDDATE(ngày, số_ngày)
DATE_ADD(ngày, số_ngày)
```

  - Cú pháp 2:

```
ADDDATE(ngày, INTERVAL giá_trị kiểu)
DATE_ADD(ngày, INTERVAL giá_trị kiểu)
```

```
SELECT ADDDATE('2007-12-13', 31)
```

ADDDATE('2007-12-13', 31)
2008-01-13

```
ELECT ADDDATE('2007-12-13',
INTERVAL 31 DAY)
```

ADDDATE('2007-12-13', INTERVAL 31 DAY)
2008-01-13

Các hàm xử lý thời gian

- Hàm **SUBDATE()** và **DATE_SUB():** có cùng kết quả trả về là một ngày mới sau khi đã trừ đi một đơn vị thời gian
  - Cách sử dụng và cú pháp của hai hàm này tương tự như hai hàm **ADDDATE()** và **DATE_ADD()**

```
SELECT SUBDATE('2007-12-13
23:59:59', 31)
```

SUBDATE('2007-12-13 23:59:59', 31)
2007-11-12 23:59:59

```
SELECT DATE_SUB(CURDATE(),
INTERVAL 30 DAY)
```

DATE_SUB(CURDATE(),INTERVAL 30 DAY)
2007-11-13

Các hàm xử lý thời gian

- Hàm **CURDATE(), CURRENT_DATE()** có kết quả trả về là ngày hiện hành của hệ thống
  - Cú pháp:

```
CURDATE()
CURRENT_DATE()
```

- Hàm **CURTIME(), CURRENT_TIME()** có kết quả trả về là giờ hiện hành của hệ thống.
  - Cú pháp:

```
CURTIME()
CURRENT_TIME()
```

Các hàm xử lý thời gian

- Hàm **NOW():** có kết quả trả về là ngày giờ hiện hành của hệ thống
  - Cú pháp:

NOW()

```
SELECT NOW()
```

- Hàm DATE() có kết quả trả về là ngày-tháng-năm của một biểu thức thời gian bất kỳ
  - Cú pháp:

DATE(biểu thức thời gian)

```
SELECT DATE('2007-12-14 07:58:59')
```

DATE('2007-12-14 07:58:59')
2007-12-14

Các hàm xử lý thời gian

- Hàm **MONTH**() có kết quả trả về là tháng của một biểu thức thời gian bất kỳ
  - Cú pháp:

MONTH(biểu thức thời gian)

```
SELECT MONTH('2007-12-13
08:03:20')
```

MONTH('2007-12-13 08:03:20')
12

- Hàm **MONTHNAME()** có kết quả trả về là tên của tháng (tiếng Anh) của của một biểu thức thời gian bất kỳ
  - Cú pháp:

MONTHNAME(biểu thức thời gian)

```
SELECT MONTHNAME(NOW())
```

MONTHNAME(NOW())
December

Các hàm xử lý thời gian

- Hàm YEAR() có kết quả trả về là năm của một biểu thức thời gian bất kỳ
  - Cú pháp

YEAR(biểu thức thời gian)

```
SELECT YEAR('2007-12-14
08:13:40')
```

YEAR('2007-12-14 08:13:40')
2007

- Hàm **DAY()** và DAYOFMONTH() có kết quả trả về là giá trị ngày của một biểu thức thời gian có kiểu ngày/ngày giờ bất kỳ
  - Cú pháp:

```
SELECT DAY(NOW())
```

DAY(biểu thức thời gian)

DAYOFMONTH(biểu thức thời gian)

```
SELECT DAYOFMONTH('2021-
08-124 10:10:10')
```

Các hàm xử lý thời gian

- Hàm **DAYNAME()** có kết quả trả về là tên của ngày trong tuần của một biểu thức thời gian có kiểu ngày/ngày giờ bất kỳ
  - Cú pháp:

DAYNAME(biểu thức thời gian)

```
SELECT DAYNAME('2007-12-24 23:59:59')
```

DAYNAME('2007-12-24 23:59:59')
Monday

- Hàm DAYOFWEEK(): kết quả trả về là giá trị số tương ứng với ngày trong tuần
  - Kết quả trả về từ 1->7, trong đó 1 tương ứng với 'Sunday', 2 tương ứng với 'Monday', …
  - Cú pháp:

DAYOFWEEK('2007-12-24 23:59:59')
2

DAYOFWEEK(biểu thức thời gian)

Các hàm xử lý thời gian

- Hàm **DAYOFYEAR()** có kết quả trả về là ngày trong năm của một biểu thức thời gian có kiểu ngày/ngày giờ bất kỳ
  - Cú pháp:

DAYOFYEAR(biểu thức thời gian)

```
SELECT DAYOFYEAR(
'2007-12-24 23:59:59')
```

DAYOFYEAR('2007-12-24 23:59:59')
358

Các hàm xử lý thời gian

- Các hàm **SECOND(), MINUTE(), HOUR(), TIME()** có kết quả trả về là một số nguyên chỉ định giây, phút, giờ và thời gian của một biểu thức thời gian có kiểu giờ:phút:giây hoặc kiểu ngày-thángnăm giờ:phút:giây bất kỳ
  - Cú pháp:

```
SECOND(biểu thức thời gian)
MINUTE(biểu thức thời gian)
HOUR(biểu thức thời gian)
TIME(biểu thức thời gian)
```

```
SELECT SECOND(NOW())
```

SECOND(NOW())
40

```
SELECT MINUTE(NOW())
```

MINUTE(NOW())
4

```
SELECT HOUR('2007-12-14 09:06:15')
```

HOUR('2007-12-14 09:06:15')
9

```
SELECT TIME(NOW())
```

TIME(NOW())
09:07:46

Các hàm xử lý thời gian

- Hàm **DATEDIFF**: có kết quả trả về là khoảng cách đại số giữa hai ngày bất kỳ
  - Cú pháp:

DATEDIFF(expr1,expr2)

  - Chú ý: Nếu expr1 nhỏ hơn expr2 thì kết quả sẽ là số nguyên âm, ngược lại thì kết quả sẽ là số nguyên dương

```
SELECT orderNumber,
DATEDIFF(requiredDate,shippedDate) AS daysLeft
FROM orders
ORDER BY daysLeft DESC;
```

orderNumber	daysLeft
10409	11
10410	10
10419	9
10398	9

Các hàm xử lý thời gian

- Hàm **TIMEDIFF()** có kết quả trả về là khoảng cách đại số của hai biểu thức thời gian bất kỳ
  - Cú pháp:

`TIMEDIFF(expr1, expr2)`

```
SELECT TIMEDIFF('23:59:59', '10:44:45')
```

TIMEDIFF('23:59:59', '10:44:45')
13:15:14

Các hàm xử lý thời gian

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

Ví dụ 1: đưa ra ngày tháng sau ngày giờ hiện tại 30 ngày:

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

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

Ví dụ 2: đưa ra các đơn đặt hàng trong khoảng 30 ngày tính từ ngày 1/5/2005

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

orderNumber	orderDate	requiredDate	shippedDate	status	comments
10411	2005-05-01 00:00:00	2005-05-08 00:00:00	2005-05-06 00:00:00	Shipped	NULL
10412	2005-05-03 00:00:00	2005-05-13 00:00:00	2005-05-05 00:00:00	Shipped	NULL
10413	2005-05-05 00:00:00	2005-05-14 00:00:00	2005-05-09 00:00:00	Shipped	Customer requested that DHL is used for this shipping

Các hàm xử lý thời gian

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

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

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

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

Ví dụ 2: đưa ra các đơn hàng đặt năm 2005

```
SELECT *
FROM orders
WHERE EXTRACT(YEAR FROM orderDate) = 2005
```

orderNumber	orderDate	requiredDate	shippedDate	status	comments
10411	2005-05-01 00:00:00	2005-05-08 00:00:00	2005-05-06 00:00:00	Shipped	NULL
10412	2005-05-03 00:00:00	2005-05-13 00:00:00	2005-05-05 00:00:00	Shipped	NULL
10413	2005-05-05 00:00:00	2005-05-14 00:00:00	2005-05-09 00:00:00	Shipped	Customer requested that DHL is used for this shipping
10414	2005-05-06 00:00:00	2005-05-13 00:00:00	NULL	On Hold	Customer credit limit exceeded. Will ship when a payment is received.

Hàm LAST_INSERT_ID

- Hàm **LAST_INSERT_ID** trả về ID của bản ghi cuối cùng được chèn vào bảng, với điều kiện đó là ID của cột có thuộc tính **AUTO_INCREMENT**

```
SELECT LAST_INSERT_ID();
```

LAST_INSERT_ID()
1

- MySQL **LAST_INSERT_ID** hoạt động dựa trên nguyên tắc độc lập với client

Truy vấn nhóm

- **Các hàm thống kê**
- **Mệnh đề GROUP BY**
- **Mệnh đề HAVING**

Các hàm thống kê

❖ Hàm SUM

- Hàm trả về tổng các giá trị theo nhóm

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

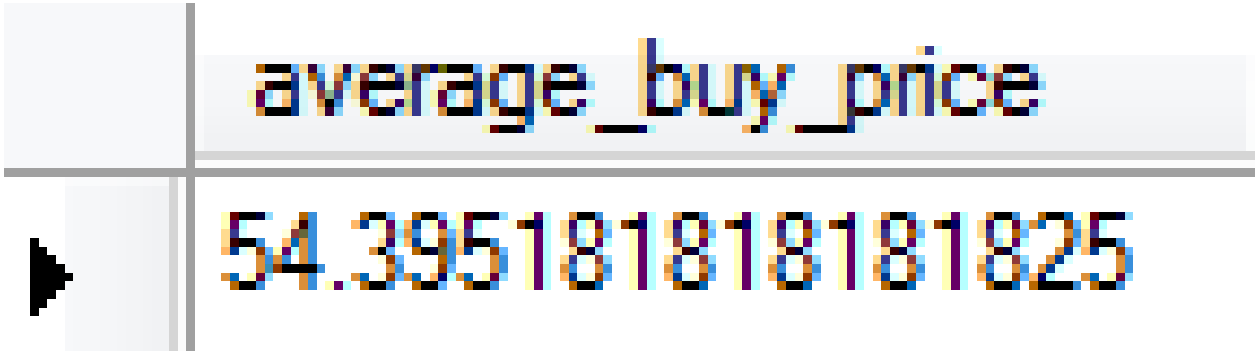
sum(quantityInStock)
555131

Các hàm thống kê

❖ Hàm AVG

- Tính giá trị trung bình của một biểu thức, Nó không chấp nhận giá trị NULL.

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

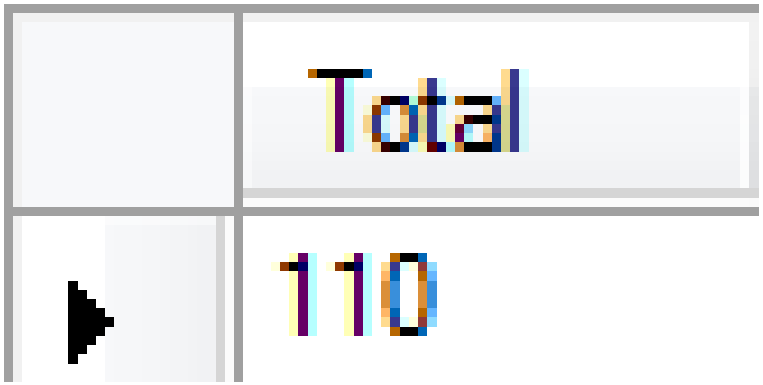


Các hàm thống kê

❖ Hàm COUNT

- Hàm trả về số lượng mẩu tin theo nhóm.

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



Các hàm thống kê

❖ 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..

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

highest_price	lowest_price
103.42	15.91

Mệnh đề nhóm GROUP BY

- Nhóm 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.
- Cú pháp:

```
SELECT Danh sách các cột, các hàm thống kê(biểu thức)
FROM tên bảng
WHERE điều kiện lọc
GROUP BY Danh sách các cột nhóm dữ liệu
[ORDER BY Tên cột sắp xếp [DESC, …]]
```

Mệnh đề nhóm GROUP BY

❖ **Ví dụ**: Giả sử muốn phân chia các đơn đặt hàng theo các nhóm phụ thuộc vào tình trạng của các đơn hàng.

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

status
Cancelled
Disputed
In Process

❖ **Ví dụ 2**: Xác định có bao nhiêu đơn đặt hàng trong từng nhóm trạng thái

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

status	count(*)
Cancelled	6
Disputed	3
In Process	6

Mệnh đề điều kiện HAVING

- Một điều kiện lọc trên dữ liệu là một nhóm các bản ghi hoặc là kết quả của việc thực hiện hàm nhóm.
- **HAVING** thường được sử dụng cùng với **GROUP BY**, điều kiện lọc chỉ được áp dụng trên các cột xuất hiện trong mệnh đề **GROUP BY**.
- **HAVING** không đi kèm với **GROUP BY** có ý nghĩa là **WHERE**
- Cú pháp:

```
SELECT Danh sách các cột, Hàm thống kê [as tên]
FROM Tên bảng
[WHERE Điều kiện lọc]
GROUP BY Danh sách các cột nhóm dữ liệu
HAVING Điều kiện lọc sau khi nhóm
[ORDER BY Tên cột sắp xếp [DESC, …]
```

Mệnh đề điều kiện HAVING

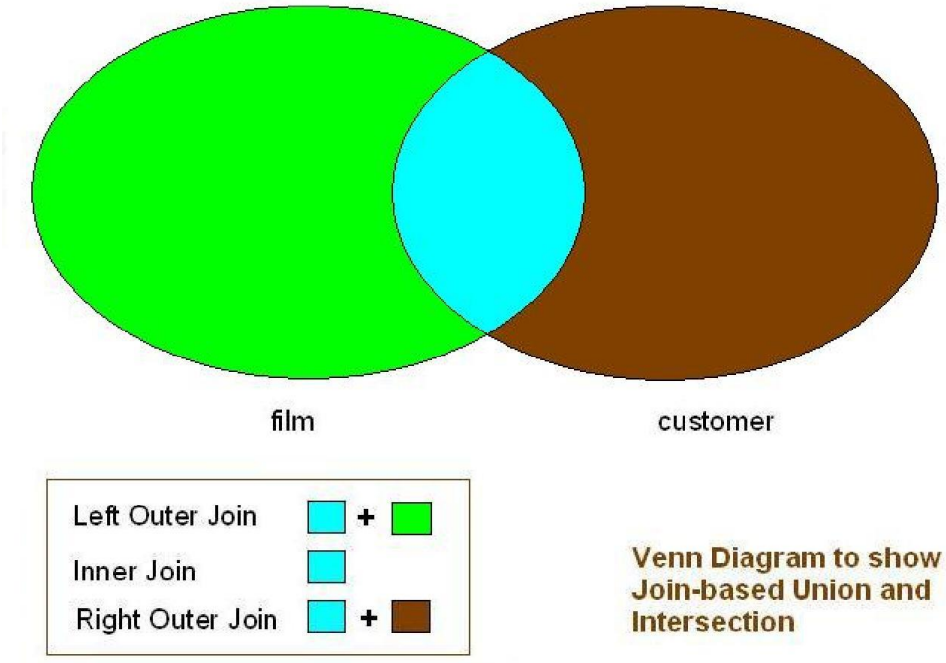
- Ví dụ: Yêu cầu chỉ lọc ra những đơn hàng có tổng giá trị lớn hơn $1000

```
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

Truy xuất dữ liệu từ nhiều bảng

- **PHÉP NỐI TRONG (INNER JOIN)**
- **PHÉP NỐI TRÁI (LEFT JOIN)**
- **PHÉP NỐI PHẢI (RIGHT JOIN)**
- **PHÉP TỰ NỐI (SELF JOIN)**



Venn Diagram to show Join-based Union and Intersection

INNER JOIN

- Chỉ định việc so sánh giá trị trong các cột của các bảng là tương đương – dữ liệu đều có ở cả hai bảng
- Kết quả sau khi thực hiện câu lệnh truy vấn kết hợp **INNER JOIN** là các mẩu tin thỏa điều kiện quan hệ ở cả hai bảng
- Cú pháp:

```
SELECT Danh sách các cột
FROM Bảng 1
INNER JOIN Bảng 2 ON điều kiện nối 1
INNER JOIN Bảng 3 ON điều kiện nối 2
...
[WHERE Điều kiện lọc]
[ORDER BY Danh sách các cột sắp xếp [DESC]];
```

INNER JOIN

❖ **Ví dụ**: Tìm ra những sản phẩm đã được bán

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

productScale VARCHAR(10) | priceEach DOUBLE

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

LEFT JOIN

- Một tùy chọn của câu lệnh SELECT cho phép lấy thêm dữ liệu từ các bảng khác.
- Chọn tất cả các hàng từ bảng bên trái ngay cả khi không có bản ghi phù hợp với nó trong bảng bên phải
- Cú pháp:

```
SELECT Danh sách các cột
FROM Bảng 1
LEFT JOIN Bảng 2 ON điều kiện nối 1
LEFT JOIN Bảng 3 ON điều kiện nối 2
...
[WHERE Điều kiện lọc]
[ORDER BY Danh sách các cột sắp xếp [DESC]];
```

LEFT JOIN

❖ **Ví dụ**: Kiểm tra tình trạng hóa đơn của các khách hàng

```
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

LEFT JOIN

❖ **Ví dụ 2**: Tìm tất cả các khách hàng không có bất kỳ đơn đặt hàng nào trong cơ sở dữ liệu

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

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

RIGHT JOIN

- Chỉ định việc so sánh giá trị trong các cột của các bảng được ưu tiên cho mối quan hệ bên nhánh phải
- Cú pháp:

```
SELECT Danh sách các cột
FROM Bảng 1
RIGHT JOIN Bảng 2 ON điều kiện nối 1
RIGHT JOIN Bảng 3 ON điều kiện nối 2
...
[WHERE Điều kiện lọc]
[ORDER BY Danh sách các cột sắp xếp [DESC]];
```

SELF JOIN

- 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: Tìm người quản lý của các nhân viên

```
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;
```

extension VARCHAR(10)

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