BÀI 1:
Quản trị CSDL MySQL

Mục tiêu bài học

- Giới thiệu các khái niệm trong RDBMS
- Giới thiệu về MySQL
- Các câu lệnh SQL thông dụng
- Truy vấn dữ liệu trong CSDL
- Liên kết các bảng trong MySQL

Giới thiệu về cơ sở dữ liệu

- Cơ sở dữ liệu (Database)
  - Là một nơi lưu trữ tập hợp các dữ liệu của ứng dụng, để người dùng có thể tìm kiếm, cập nhật, sắp xếp, truy vấn.
- Relational DataBase Management System (RDBMS) là một phần mềm trong đó:
  - Cung cấp một môi trường để có thể quản lý, tạo, sửa, quản trị các CSDL.
  - Cho phép quản lý và thao tác với dữ liệu trong các CSDL.
  - Cung cấp cơ chế bảo mật, phân quyền truy cập vào các CSDL.
- Mô hình dữ liệu quan hệ là một tập hợp các quy tắc lưu trữ, quản lý dữ liệu, trong đó:
  - Dữ liệu trong CSDL quan hệ được tổ chức thành các bảng có quan hệ với nhau.
  - Quan hệ giữa các bảng được thiết lập bằng cách sử dụng primary key/foreign key.

Giới thiệu về cơ sở dữ liệu

- Dữ liệu trong RDBMS được lưu trữ trong các đối tượng CSDL (database objects) được gọi là các bảng. Một bảng sẽ bao gồm một tập hợp các bản ghi dữ liệu và bao gồm các dòng/cột.
- Mỗi bảng sẽ bao gồm một tập hợp các cột (column). Mỗi cột còn được gọi là một field, dùng để lưu trữ một thuộc tính của các bản ghi trong bảng.
- Record, còn được gọi là một row, là một bản ghi lưu thông tin về một thực thể trong bảng.

Các thuật ngữ về RDBMS

- **Database**
  - Là một kho chứa một tập hợp các bảng, với dữ liệu có quan hệ với nhau.
- **Table**
  - Là một đơn vị lưu trữ dữ liệu trong CSDL. Mỗi bảng lưu trữ dữ liệu về một tập thực thể, bao gồm các dòng và các cột.
- **Column**
  - Còn được gọi là một field trong bảng. Mỗi cột chứa dữ liệu về một thuộc tính của bảng, VD như: mã, họ tên, địa chỉ.
- **Row**
  - Một dòng (hay còn được gọi là một tuple, entry hoặc record) là một tập hợp dữ liệu có liên quan đến nhau. Mỗi dòng sẽ chứa dữ liệu về một thực thể trong bảng.
- **Primary Key**
  - Khóa chính (primary key) là một thuộc tính của bảng, dùng để định danh mỗi bản ghi trong bảng là duy nhất và không trùng lặp. Mỗi bảng chỉ có duy nhất một khóa chính. Đặc điểm của khóa chính là: Không được trống, và không trùng lặp.

Các thuật ngữ về RDBMS

- **Foreign Key**
  - Khóa ngoại (foreign key) là một cột của một bảng có tham chiếu đến khóa chính của một bảng khác. Các giá trị được chèn vào khóa ngoại phải tồn tại trong khóa chính mà nó tham chiếu
- **Compound Key**
  - Khóa tổ hợp (compound key hoặc composite key) là một khóa được tạo nên bao gồm nhiều cột trong bảng.
- **Index**
  - Chỉ mục (index) là một thành phần của bảng, dùng để lưu lại vị trí của các bản ghi trong bảng. Index giúp tang tốc độ truy vấn dữ liệu trong bảng (tương tự như mục lục của cuốn sách).
- **Unique Key**
  - Khóa duy nhất (Unique Key) là một thuộc tính được gắn với một cột trong bảng, để đảm bảo dữ liệu được nhập vào cột đó là duy nhất, không trùng lặp.

Giới thiệu về MySQL

- MySQL là một DBMS mã nguồn mở, miễn phí được thực thi ở server, và được sử dụng trong các ứng dụng web.
- MySQL có thể phù hợp với cả các ứng dụng có quy mô từ nhỏ đến lớn.
- MySQL dễ sử dụng, tin cậy, tốc độ nhanh, tương thích với nhiều nền tảng.
- MySQL miễn phí khi download và sử dụng
- MySQL được phát triển, phân phối và hỗ trợ bởi Oracle Corporation
- Dữ liệu trong MySQL database được lưu trữ trong các bảng, và tuân theo mô hình quan hệ.

Giới thiệu về MySQL

- **MySQL** là hệ quản trị cơ sở dữ liệu tự do nguồn mở phổ biến nhất thế giới và được các nhà phát triển rất ưa chuộng trong quá trình phát triển ứng dụng.
- Vì MySQL là cơ sở dữ liệu tốc độ cao, ổn định và dễ sử dụng, có tính khả chuyển, hoạt động trên nhiều hệ điều hành cung cấp một hệ thống lớn các hàm tiện ích rất mạnh.
- Với tốc độ và tính bảo mật cao, MySQL rất thích hợp cho các ứng dụng có truy cập CSDL trên internet. Người dùng có thể tải về MySQL miễn phí từ trang chủ.

Cơ sở dữ liệu MySQL

- MySQL là một Hệ quản trị CSDL có tốc độ xử lý nhanh, dễ sử dụng, phù hợp cho nhiều loại ứng dụng từ nhỏ đến lớn.
- Một số điểm mạnh khiến MySQL trở nên phổ biến:
  - MySQL được phát hành dưới bản quyền nguồn mở, vì vậy ta có thể sử dụng mà không phải trả phí.
  - MySQL là một RDBMS tuy miễn phí nhưng rất mạnh mẽ, có thể hỗ trợ các chức năng quan trọng giống như các hệ quản trị CSDL thương mại khác.
  - MySQL làm việc tốt với nhiều hệ điều hành và với nhiều ngôn ngữ lập trình khác nhau, như PHP, PERL, C, C++, JAVA, v.v....
  - MySQL có tốc độ thực thi rất tốt, có thể xử lý nhanh chóng đối với các tập dữ liệu lớn.
  - MySQL thường được sử dụng với PHP, là ngôn ngữ lập trình web phổ biến nhất trên thế giới.

SQL CREATE DATABASE

- Lệnh CREATE DATABASE được dùng để tạo mới một CSDL.
  - Chú ý: Cần đảm bảo được cấp quyền quản trị trước khi tạo CSDL.
  - Sau khi database được tạo, ta có thể xem danh sách CSDL có trên server bằng lệnh sau: SHOW DATABASES;
- Cú pháp:

```
CREATE DATABASE databasename;
```

- Ví dụ:

```
CREATE DATABASE testDB;
```

SQL DROP DATABASE

- Lệnh DROP DATABASE được dùng để xóa một CSDL đang tồn tại.
- Chú ý: Cần thận trọng trước khi xóa CSDL. Khi ta xóa CSDL, tất cả dữ liệu và các đối tượng trong CSDL đó sẽ bị xóa !
- Cú pháp:

```
DROP DATABASE databasename;
```

- Ví dụ:

```
DROP DATABASE testDB;
```

SQL CREATE TABLE

- Lệnh CREATE TABLE được dùng để tạo một bảng mới trong database.
  - Tham số column thể hiện tên các cột trong bảng.
  - Tham số datatype thiết lập kiểu dữ liệu trong các cột của bảng (VD: varchar, integer, date, v.v...).
- Cú pháp:

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
   ....
);
```

- Ví dụ:

```
CREATE TABLE Persons (
    PersonID int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255)
);
```

SQL DROP TABLE

- Lệnh DROP TABLE được dùng để xóa một bảng trong database.
  - Chú ý: Cần thận trọng khi xóa bảng. Khi ta xóa bảng, dữ liệu trong bảng sẽ bị mất hết.
- Cú pháp:

```
DROP TABLE table_name;
```

- Ví dụ:

```
DROP TABLE Shippers;
```

- SQL TRUNCATE TABLE
  - Dùng để xóa tất cả dữ liệu trong bảng, nhưng không xóa bảng đó.

```
TRUNCATE TABLE table_name;
```

SQL ALTER TABLE

- Lệnh ALTER TABLE được dùng để sửa cấu trúc bảng. Lệnh này dùng để thêm, sửa, xóa các cột trong một bảng đã tồn tại.
- Lệnh ALTER TABLE cũng được dùng để thêm hoặc xóa các ràng buộc (constraints) trên bảng.
- Lệnh ALTER TABLE - ADD Column

```
ALTER TABLE table_name
ADD column_name datatype;
```

- Lệnh ALTER TABLE - DROP COLUMN

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

- Lệnh ALTER TABLE - ALTER/MODIFY COLUMN

```
ALTER TABLE table_name
MODIFY COLUMN column_name datatype;
```

Ràng buộc (Constraints)

- SQL constraints là những ràng buộc được dùng để thiết lập các quy tắc cho dữ liệu trong một bảng.
- Constraints được dùng để hạn chế dữ liệu được nhập vào bảng. Constraints sẽ đảm bảo tính chính xác và tin cậy của dữ liệu trong bảng.
- Nếu trong quá trình thao tác với dữ liệu trong bảng, nếu có bất kỳ một sự vi phạm các ràng buộc đã được thiết lập cho bảng, hành động sẽ bị hủy, dữ liệu sẽ không được cập nhật trong bảng.
- Constraints có thể được áp dụng cho cột (column level) hoặc cho toàn bộ bảng (table level).

Ràng buộc (Constraints)

- Một số ràng buộc phổ biến trong SQL:
  - **NOT NULL**
    - Bảo đảm rằng một cột không được để trống.
  - **UNIQUE**
    - Đảm bảo rằng tất cả giá trị trong một cột là duy nhất.
  - **PRIMARY KEY**
    - Là một ràng buộc được áp dụng cho toàn bộ bảng để đảm bảo mỗi dòng trong bảng là duy nhất.
  - **FOREIGN KEY**
    - Là khóa ngoại có tham chiếu đến khóa chính của một bảng khác.
  - **CHECK**
    - Là một ràng buộc được áp dụng để đảm bảo rằng giá trị được chèn vào cột sẽ phải thỏa mãn những điều kiện của người dùng.
  - **DEFAULT**
    - Cho phép thiết lập một giá trị mặc định cho cột, khi ta không chèn dữ liệu cho cột đó.
  - **INDEX**
    - Cho phép tăng tốc độc truy vấn dữ liệu trong bảng.

AUTO INCREMENT

- MySQL hỗ trợ cơ chế tự sinh cho những cột cần dữ liệu duy nhất, hoặc tự động sinh dữ liệu khi ta cần chèn một bản ghi mới vào bảng, thông thường là khóa chính.
- Auto-increment cho phép tự động sinh một số duy nhất khi một bản ghi mới được chèn vào bảng.
- MySQL sử dụng từ khóa AUTO_INCREMENT để cài đặt cơ chế tự sinh.
- Mặc định, giá trị khởi tạo cho thuộc tính AUTO_INCREMENT là 1, và sau mỗi lần chèn một bản ghi mới, thuộc tính này sẽ tăng lên 1.
- Ví dụ:

```
CREATE TABLE Persons (
    ID int NOT NULL AUTO_INCREMENT,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    PRIMARY KEY (ID)
);
```

SQL INSERT INTO

- Lệnh INSERT INTO được dùng để chèn một bản ghi mới vào bảng.
- Có 2 cách để viết lệnh INSERT INTO để chèn dữ liệu vào bảng.
  - Cách 1: Chỉ định cả tên các cột và giá trị cần chèn vào bản ghi mới

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

  - Cách 2: Nếu ta chèn dữ liệu vào tất cả các cột trong bảng, ta không cần phải liệt kê tên các cột trong câu lệnh INSERT. Tuy nhiên, ta cần đảm bảo thứ tự các giá trị cần chèn sẽ phải đúng với thứ tự các cột trong bảng

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

SQL UPDATE

- Lệnh UPDATE được dùng để sửa đổi dữ liệu trong bảng.
  - Chú ý: Cần có mệnh đề WHERE khi tiến hành update dữ liệu. Nếu không có mệnh đề WHERE, tất cả các bản ghi trong bảng sẽ được cập nhật.
- Cú pháp:

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

- Ví dụ:

```
UPDATE Customers
SET ContactName='Juan'
WHERE Country='Mexico';
```

SQL DELETE

- Lệnh DELETE được dùng để xóa dữ liệu trong bảng.
  - Chú ý: Cần có mệnh đề WHERE khi tiến hành xóa dữ liệu. Nếu không có mệnh đề WHERE, tất cả các bản ghi trong bảng sẽ **bị xóa.**
- Cú pháp:

```
DELETE FROM table_name
WHERE condition;
```

- Ví dụ:

```
DELETE FROM Customers
WHERE CustomerName='Alfreds Futterkiste';
```

Thực hành

- GV hướng dẫn sử dụng MySQL
- Sinh viên thực hành sử dụng MySQL: Tạo CSDL, tạo bảng

Giới thiệu về SQL

- SQL là một ngôn ngữ chuẩn để truy vấn và thao tác dữ liệu trong các cơ sở dữ liệu.
- SQL là gì ?
  - SQL viết tắt của Structured Query Language (ngôn ngữ truy vấn có cấu trúc)
  - SQL cung cấp các câu lệnh truy vấn cho phép truy cập và thao tác với dữ liệu trong các CSDL.
  - SQL là một chuẩn do Viện tiêu chuẩn quốc gia Hoa Kỳ American National Standards Institute (ANSI) đưa ra năm 1986, và là một chuẩn của International Organization for Standardization (ISO) vào năm 1987.
- Để xây dựng một web site có chứa dữ liệu trong một database, ta cần những công nghệ sau:
  - Một ứng dụng RDBMS (VD như: MS Access, SQL Server, MySQL)
  - Một ngôn ngữ lập trình kịch bản phía server (server-side scripting language), VD như PHP, JSP hoặt ASP.NET
  - Ngôn ngữ SQL để truy vấn và cập nhật dữ liệu trong CSDL.
  - Sử dụng HTML / CSS để tạo các trang web

SQL SELECT

- Lệnh SELECT dùng để select dữ liệu từ database.
- Tập hợp dữ liệu được trả về từ câu lệnh select sẽ là một tập hợp các dòng và các cột theo dạng bảng, còn được gọi là result-set.
- Cú pháp:

```
SELECT column1, column2, ...
FROM table_name;
```

- Danh sách các tên cột là những tên cột mà ta muốn lấy dữ liệu trong bảng. Nếu ta muốn lấy tất cả dữ liệu trong các cột của bảng, ta sử dụng cú pháp sau:

```
SELECT * FROM table_name;
```

Mệnh đề WHERE

- Mệnh đề WHERE được dùng để lọc các bản ghi được truy vấn trong bảng.
- Mệnh đề WHERE được dùng để thiết lập các tiêu chí lọc, chỉ lấy ra những bản ghi trong bảng thỏa mãn điều kiện tìm kiếm.
- Mệnh đề WHERE không chỉ được sử dụng cho câu lệnh SELECT, mà nó còn được sử dụng cho các câu lệnh UPDATE, DELETE.
- Cú pháp:

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

- Ví dụ

```
SELECT * FROM Customers
WHERE Country='Mexico';
```

SQL SELECT DISTINCT

- Lệnh SELECT DISTINCT dùng để truy vấn những giá trị duy nhất (không trùng lặp) trong bảng.
- Trong bảng, một cột có thể chứa nhiều giá trị trùng lặp, nếu truy vấn tất cả thì sẽ mất nhiều thời gian và tài nguyên, vì vậy nếu ta chỉ muốn lấy ra những giá trị duy nhất trong bảng, ta cần dùng mệnh đề DISTINCT.

- Cú pháp:

```
SELECT DISTINCT column1, column2, ...
FROM table_name;
```

Mệnh đề WHERE

- Dưới đây là danh sách các toán tử mà mệnh đề WHERE hỗ trợ:

Toán tử	Ý nghĩa
=	So sánh bằng
<> hoặc !=	So sánh không bằng
>	So sánh lớn hơn
<	So sánh nhỏ hơn
>=	So sánh lớn hơn hoặc bằng
<=	So sánh nhỏ hơn hoặc bằng
BETWEEN	Tìm kiếm trong một khoảng giá trị
LIKE	Tìm kiếm tương đối theo một mẫu nào đó
IN	Tìm kiếm trong một danh sách các giá trị được liệt kê

Toán tử AND, OR và NOT

- Mệnh đề WHERE có thể được kết hợp với các toán tử AND, OR, và NOT.
- Các toán tử logic AND và OR được sử dụng để lọc các bản ghi dựa trên các điều kiện sau:
  - Toán tử AND sẽ chọn bản ghi nếu các điều kiện được chỉ định bởi toán tử AND đều là TRUE.
  - Toán tử OR sẽ chọn bản ghi nếu một trong số các điều kiện được chỉ định bởi toán tử OR là TRUE.
- Toán tử NOT sẽ tìm kiếm những bản ghi nào thỏa mãn điều kiện phủ định.

Toán tử AND, OR và NOT

- Cú pháp toán tử AND

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 AND condition2 AND condition3 ...;
```

- Cú pháp toán tử OR

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 OR condition2 OR condition3 ...;
```

- Cú pháp toán tử NOT

```
SELECT column1, column2, ...
FROM table_name
WHERE NOT condition;
```

SQL ORDER BY

- Mệnh đề ORDER BY được dùng để sắp xếp tập kết quả được tìm kiếm trong bảng theo thứ tự tăng dần hoặc giảm dần.
- Mặc định, thứ tự sắp xếp của mệnh đề ORDER BY sẽ là tăng dần. Để sắp xếp theo thứ tự giảm dần, ta sử dụng từ khóa DESC.
- Cú pháp:

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;
```

Mệnh đề SELECT TOP

- Mệnh đề SELECT TOP được dùng để chỉ định số bản ghi được truy vấn.
- Mệnh đề SELECT TOP được dùng cho những bảng có khối lượng dữ liệu lớn, từ hang ngàn bản ghi trở lên. Việc truy vấn và trả về một số lượng lớn các bản ghi sẽ ảnh hưởng đến hiệu suất của ứng dụng, vì vậy việc sử dụng mệnh đề SELECT TOP sẽ hữu ích.
- **Chú ý:** Không phải tất cả các DBMS đều hỗ trợ mệnh đề SELECT TOP. MySQL hỗ trợ mệnh đề LIMIT để truy vấn một số lượng hữu hạn các bản ghi records, trong khi đó Oracle lại sử dụng lệnh ROWNUM.
- Cú pháp:

```
SELECT column_name(s)
FROM table_name
WHERE condition
LIMIT number;
```

Toán tử LIKE

- Toán tử LIKE được sử dụng cùng với mệnh đề WHERE để tìm kiếm tương đối theo một khuôn mẫu (pattern) trong một cột.
  - Thông thường toán tử LIKE được dùng để tìm kiếm trong những cột với kiểu dữ liệu ký tự.
- Có 2 wildcards được dùng kết hợp với toán tử LIKE:
  - % - Đại diện cho 0, 1 hoặc nhiều ký tự.
  - _ - Đại diện cho một ký tự đơn.
- Ta có thể phối hợp thêm các điều kiện trong toán tử LIKE bằng cách sử dụng các toán tử AND, OR hoặc NOT.
- Cú pháp:

```
SELECT column1, column2, ...
FROM table_name
WHERE columnN LIKE pattern;
```

Ví dụ

- Dưới đây là một số ví dụ về việc sử dụng toán tử LIKE với các wildcards '%' và '_' :

LIKE Operator	Description
WHERE CustomerName LIKE 'a%'	Finds any values that starts with "a"
WHERE CustomerName LIKE '%a'	Finds any values that ends with "a"
WHERE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'a_%_%'	Finds any values that starts with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%o'	Finds any values that starts with "a" and ends with "o"

Toán tử IN

- Toán tử IN cho phép liệt kê nhiều giá trị để so sánh trong mệnh đề WHERE.
- Toán tử IN là một dạng cú pháp viết tắt của nhiều điều kiện OR.
- Cú pháp:

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (value1, value2, ...);
```

- Ví dụ:

```
SELECT * FROM Customers
WHERE Country IN ('Germany', 'France', 'UK');
```

Toán tử BETWEEN

- Toán tử BETWEEN sẽ lựa chọn các giá trị ở trong một miền nhất định. Các giá trị có thể thuộc kiểu dữ liệu số, văn bản hoặc ngày tháng.
- Toán tử BETWEEN sẽ bao gồm cả so sánh bằng.
- Cú pháp:

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

- Ví dụ:

```
SELECT * FROM Products
WHERE Price BETWEEN 10 AND 20;
```

SQL MIN() và MAX()

- Hàm MIN() trả về giá trị nhỏ nhất trong cột được chọn.
- Hàm MAX() trả về giá trị lớn nhất trong cột được chọn .
- Cú pháp hàm MIN()

```
SELECT MIN(column_name)
FROM table_name
WHERE condition;
```

- Cú pháp hàm MAX()

```
SELECT MAX(column_name)
FROM table_name
WHERE condition;
```

COUNT(),AVG(),SUM()

- Hàm COUNT() trả về số dòng được tìm thấy dựa trên một điều kiện nào đó.
- Hàm AVG() trả về giá trị trung bình cộng của một cột có giá trị số.
- Hàm SUM() trả về tổng giá trị của một cột kiểu số.
- Cú pháp:

```
SELECT COUNT(column_name)
FROM table_name
WHERE condition;
```

```
SELECT AVG(column_name)
FROM table_name
WHERE condition;
```

```
SELECT SUM(column_name)
FROM table_name
WHERE condition;
```

SQL Aliases

- SQL aliases are used to give a table, or a column in a table, a temporary name.
- Aliases are often used to make column names more readable.
- An alias only exists for the duration of the query.
  - **Note:** It requires double quotation marks or square brackets if the alias name contains spaces
- Syntax

```
SELECT column_name AS alias_name
FROM table_name;
```

- Example

```
SELECT CustomerID as ID, CustomerName AS Customer
FROM Customers;
```

SQL NULL Values

- A field with a NULL value is a field with no value.
- If a field in a table is optional, it is possible to insert a new record or update a record without adding a value to this field. Then, the field will be saved with a NULL value.
  - It is very important to understand that a NULL value is different from a zero value or a field that contains spaces. A field with a NULL value is one that has been left blank during record creation
- It is not possible to test for NULL values with comparison operators, such as =, <, or <>.

IS NOT NULL

```
SELECT column_names
FROM table_name
WHERE column_name IS NULL;
```

SQL Data Types

- Các cột trong bảng sẽ chứa dữ liệu thuộc một kiểu dữ liệu. Một kiểu dữ liệu sẽ định nghĩa loại dữ liệu mà cột đó có thể chứa: dữ liệu số nguyên, số thức, ký tự, tiền tệ, ngày tháng, nhị phân v.v…
- Khi tạo bảng, người dùng cần quyết định xem loại dữ liệu mà từng cột trong bảng chứa sẽ là loại nào.
- Tùy từng cột mà người dùng sẽ quyết định kiểu dữ liệu, cũng như kích thước dữ liệu của cột đó
  - Chú ý: Kiểu dữ liệu có thể sẽ khác nhau, tùy thuộc vào từng loại CSDL.

MySQL Data Types

- Trong CSDL MySQL có nhiều kiểu dữ liệu, nhưng được phân thành 3 nhóm chính: text, number, và date.
- Các kiểu dữ liệu text:

Data type	Description
CHAR(size)	Holds a fixed length string (can contain letters, numbers, and special characters). The fixed size is specified in parenthesis. Can store up to 255 characters
VARCHAR(size)	Holds a variable length string (can contain letters, numbers, and special characters). The maximum size is specified in parenthesis. Can store up to 255 characters. **Note:** If you put a greater value than 255 it will be converted to a TEXT type
TINYTEXT	Holds a string with a maximum length of 255 characters
TEXT	Holds a string with a maximum length of 65,535 characters
BLOB	For BLOBs (Binary Large OBjects). Holds up to 65,535 bytes of data
MEDIUMTEXT	Holds a string with a maximum length of 16,777,215 characters
MEDIUMBLOB	For BLOBs (Binary Large OBjects). Holds up to 16,777,215 bytes of data
LONGTEXT	Holds a string with a maximum length of 4,294,967,295 characters
LONGBLOB	For BLOBs (Binary Large OBjects). Holds up to 4,294,967,295 bytes of data
ENUM(x,y,z,etc.)	Let you enter a list of possible values. You can list up to 65535 values in an ENUM list. If a value is inserted that is not in the list, a blank value will be inserted. **Note:** The values are sorted in the order you enter them. You enter the possible values in this format: ENUM('X','Y','Z')
SET	Similar to ENUM except that SET may contain up to 64 list items and can store more than one choice

MySQL Data Types

- Các kiểu dữ liệu number

Data type	Description
TINYINT(size)	-128 to 127 normal. 0 to 255 UNSIGNED*. The maximum number of digits may be specified in parenthesis
SMALLINT(size)	-32768 to 32767 normal. 0 to 65535 UNSIGNED*. The maximum number of digits may be specified in parenthesis
MEDIUMINT(size)	-8388608 to 8388607 normal. 0 to 16777215 UNSIGNED*. The maximum number of digits may be specified in parenthesis
INT(size)	-2147483648 to 2147483647 normal. 0 to 4294967295 UNSIGNED*. The maximum number of digits may be specified in parenthesis
BIGINT(size)	-9223372036854775808 to 9223372036854775807 normal. 0 to 18446744073709551615 UNSIGNED*. The maximum number of digits may be specified in parenthesis
FLOAT(size,d)	A small number with a floating decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter
DOUBLE(size,d)	A large number with a floating decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter
DECIMAL(size,d)	A DOUBLE stored as a string , allowing for a fixed decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter

MySQL Data Types

- Các kiểu dữ liệu date

Data type	Description
DATE()	A date. Format: YYYY-MM-DD **Note:** The supported range is from '1000-01-01' to '9999-12-31'
DATETIME()	*A date and time combination. Format: YYYY-MM-DD HH:MI:SS **Note:** The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'
TIMESTAMP()	*A timestamp. TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD HH:MI:SS **Note:** The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC
TIME()	A time. Format: HH:MI:SS **Note:** The supported range is from '-838:59:59' to '838:59:59'
YEAR()	A year in two-digit or four-digit format. **Note:** Values allowed in four-digit format: 1901 to 2155. Values allowed in two-digit format: 70 to 69, representing years from 1970 to 2069

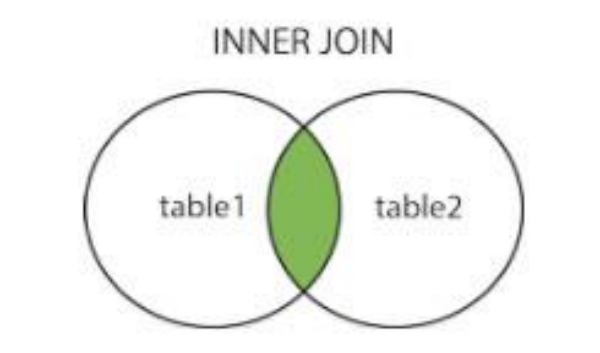
SQL JOIN

- Một mệnh đề JOIN được dùng để kết hợp và truy vấn dữ liệu từ hai hay nhiều bảng, dựa trên một cột chung giữa chúng.
- Một số loại lệnh SQL JOINs
  - **(INNER) JOIN**: Trả về các bản ghi có giá trị khớp trong cả 2 bảng.
  - **LEFT (OUTER) JOIN**: Trả về tất cả các bản ghi ở bảng bên trái, và những bản ghi thỏa mãn điều kiện (có giá trị khớp) trong bảng bên phải.
  - **RIGHT (OUTER) JOIN**: Trả về tất cả các bản ghi ở bảng bên phải, và những bản ghi thỏa mãn điều kiện (có giá trị khớp) trong bảng bên trái.
  - **FULL (OUTER) JOIN**: Trả về tất cả những bản ghi thỏa mãn điều kiện ở một trong hai bảng bên trái hoặc bên phải.

SQL INNER JOIN

- Mệnh đề INNER JOIN sẽ lựa chọn những bản ghi có giá trị khớp trong cả 2 bảng.
- Cú pháp:

```
SELECT column_name(s)
FROM table1
INNER JOIN table2 ON table1.column_name = table2.column_name;
```



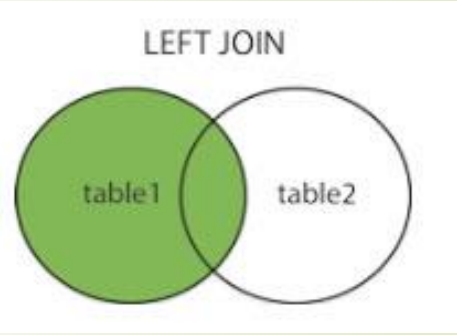
- Ví dụ:

```
SELECT Orders.OrderID, Customers.CustomerName
FROM Orders
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```

SQL LEFT JOIN

- Mệnh đề LEFT JOIN sẽ trả về tất cả các bản ghi ở bảng bên trái (table1), và những bản ghi thỏa mãn điều kiện ở bảng bên phải (table2). Nếu những dòng nào không khớp ở bảng bên phải, kết quả trả về sẽ là NULL.
- Cú pháp:

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2 ON table1.column_name = table2.column_name;
```



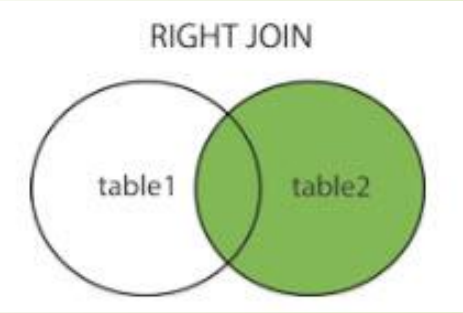
- Ví dụ:

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
ORDER BY Customers.CustomerName;
```

SQL RIGHT JOIN

- Mệnh đề RIGHT JOIN sẽ trả về tất cả các bản ghi ở bảng bên phải (table2), và những bản ghi thỏa mãn điều kiện ở bảng bên trái (table1). Nếu những dòng nào không khớp ở bảng bên trái, kết quả trả về sẽ là NULL.
- Cú pháp:

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2 ON table1.column_name = table2.column_name;
```



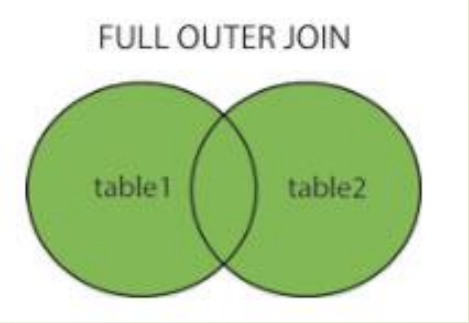
- Ví dụ:

```
SELECT Orders.OrderID, Employees.LastName, Employees.FirstName
FROM Orders
RIGHT JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID
ORDER BY Orders.OrderID;
```

FULL OUTER JOIN

- Mệnh đề FULL OUTER JOIN trả về tất cả những bản ghi mà chỉ cần khớp với một trong hai bảng bên trái (table1) hoặc bên phải (table2).
  - **Chú ý:** Việc sử dụng mệnh đề FULL OUTER JOIN có thể trả về một tập bản ghi rất lớn !
- Cú pháp:

```
SELECT column_name(s)
FROM table1
FULL OUTER JOIN table2 ON table1.column_name = table2.column_name;
```



- Ví dụ:

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
FULL OUTER JOIN Orders ON Customers.CustomerID=Orders.CustomerID
ORDER BY Customers.CustomerName;
```

Demo

- Giáo viên demo việc sử dụng câu lệnh join cho sinh viên.

Dư thừa dữ liệu (Data Redundancy)

- Dư thừa (Redundancy):
  - Làm tăng thời gian cập nhật (thêm sửa xóa) dữ liệu trong bảng.
  - Làm tăng dung lượng đĩa cứng để lưu trữ dữ liệu, do đó sẽ làm tốn tài nguyên.
- Dư thừa có thể dẫn đến những bất lợi sau:
  - Thêm sửa xóa dữ liệu, có thể dẫn đến tình trạng không nhất quán về dữ liệu.
  - Có thể có những lỗi xảy ra khi dữ liệu bị dư thừa hoặc trùng lặp.
  - Tốn kém dung lượng đĩa cứng để lưu trữ dữ liệu thừa.
  - Làm tốn thời gian truy vấn dữ liệu trong bảng.

Dư thừa dữ liệu (Data Redundancy)

- Bảng STUDENT chứa dữ liệu như hình bên dưới:

STUDENT ID	STUDENT NAME	....	STUDENT SEMESTER	STUDENT TEST1	STUDENT TEST2
001	Mary	....	SEM-1	40	65
001	Mary	....	SEM-2	56	48
002	Jake	....	SEM-1	93	84
002	Jake	....	SEM-2	85	90

Trong bảng Student, các thông tin của sinh viên như STUDENTID và STUDENTNAME bị lặp lại khi lưu lại điểm trong các học kỳ khác nhau.

Chuẩn hóa dữ liệu

- Chuẩn hóa (Normalization):
  - Là quá trình tách các bảng có cấu trúc phức tạp trở thành những bảng có cấu trúc đơn giản hơn bằng cách sử dụng một số quy luật nhất định.
  - Lợi ích của quá trình chuẩn hóa:
    - Giúp duy trì tính toàn vẹn của dữ liệu.
    - Giúp làm đơn giản hóa cấu trúc bảng, nhờ đó giúp làm cho CSDL trở nên tối ưu hơn.
    - Giúp loại bỏ và hạn chế các giá trị null, vì vậy làm giảm sự phức tạo của các thao tác dữ liệu.

Chuẩn hóa dữ liệu

- Một số quy tắc cần tuân thủ để có được một CSDL tốt:
  - Mỗi bảng cần có một khóa chính.
  - Mỗi bảng nên lưu trữ dữ liệu về một kiểu thực thể đơn..
  - Cần tránh các cột chấp nhận giá trị NULL.
  - Cần tránh lưu trữ các giá trị trùng lặp.

Chuẩn hóa dữ liệu

- Quá trình chuẩn hóa sẽ dẫn đến việc cấu trúc các bảng trong CSDL thỏa mãn một số dạng nhất định.
- Những dạng chuẩn này sẽ giúp loại bỏ đi những bất thường và không nhất quán về dữ liệu trong các bảng thuộc CSDL.
- Dưới đây là một số dạng chuẩn phổ biến:
  - Dạng chuẩn 1 (1NF)
  - Dạng chuẩn 2 (2NF)
  - Dạng chuẩn 3 (3NF)
  - Dạng chuẩn Boyce-Codd (BCNF)

Chuẩn hóa dữ liệu

- Dạng chuẩn 1 (1NF):
  - Một bảng được coi là đạt dạng chuẩn 1 khi mỗi ô trong bảng chỉ chứa một giá trị nguyên tố.
  - Một số hướng dẫn để chuyển đổi một bảng thành dạng chuẩn 1:
    - Đặt những giá trị dữ liệu có liên quan đến nhau vào trong một bảng.
    - Không đặt những nhóm dữ liệu trùng lặp trong bảng.
    - Mỗi bảng sẽ phải có một khóa chính.

Chuẩn hóa dữ liệu

- Dạng chuẩn 2 (2NF):
  - Một bảng được coi là đạt dạng chuẩn 2NF khi:
    - Bảng đó đã đạt dạng chuẩn 1NF
    - Không tồn tại sự phụ thuộc một phần giữa các thuộc tính không khóa (non-key attributes) với các thuộc tính khóa (key attributes).
  - Một số hướng dẫn để chuyển một bảng thành dạng chuẩn 2NF:
    - Tìm và loại bỏ những thuộc tính chỉ phụ thuộc hàm một phần vào khóa, tách chúng ra một bảng khác.
    - Nhóm những thuộc tính còn lại.

Chuẩn hóa dữ liệu

- Dạng chuẩn 3 (3NF):
  - Một bảng được coi là đạt dạng chuẩn 3 nếu và chỉ nếu:
    - Nó đạt dạng chuẩn 2
    - Không có sự phụ thuộc bắc cầu giữa các thuộc tính không khóa (non-key attributes) và các thuộc tính khóa (key attributes).
  - Những hướng dẫn để chuyển một bảng thành dạng chuẩn 3NF:
    - Tìm và loại bỏ những thuộc tính không khóa (non-key attributes) phụ thuộc hàm vào những thuộc tính không phải là khóa chính. Đặt chúng vào một bảng khác.
    - Nhóm những thuộc tính còn lại.

Chuẩn hóa dữ liệu

- Dạng chuẩn Boyce-Codd (BCNF):
  - Ba dạng chuẩn đôi khi không đủ để chuẩn hóa toàn bộ CSDL trong một số tình huống. Ba dạng chuẩn sẽ không thỏa mãn đối với các bảng có những đặc điểm sau:
    - Có nhiều candidate keys.
    - Có nhiều candidate keys là tổ hợp
    - Có nhiều candidate keys chồng lắp nhau (có ít nhất một thuộc tính chung).
  - Một số gợi ý để chuyển đổi một bảng thành dạng chuẩn BCNF gồm:
    - Tìm và loại bỏ những candidate keys chồng lắp nhau. Đặt các candidate key và những thuộc tính mà phụ thuộc hàm vào khóa ra một bảng khác.
    - Gom những thuộc tính còn lại vào một bảng.

Tổng kết

- Giới thiệu các khái niệm trong RDBMS
- Giới thiệu về MySQL
- Các câu lệnh SQL thông dụng
- Truy vấn dữ liệu trong CSDL
- Liên kết các bảng trong MySQL