THIẾT KẾ HỆ CSDL QUAN HỆ - XÂY DỰNG CSDL VẬT LÝ

Nguyễn Đình Hóa

dinhhoa@gmail.com

094-280-7711

B5. Xây dựng CSDL mức Vật lý

- Sử dụng ngôn ngữ định nghĩa dữ liệu để xây dựng các bảng dữ liệu, các mối liên kết giữa các bảng, các ràng buộc về mặt dữ liệu.
- Sử dụng ngôn ngữ thao tác dữ liệu để thực hiện các thao tác thêm, sửa, xoá, và truy vấn dữ liệu

SQL - MySQL

- Structural Query Language (SQL) là ngôn ngữ dữ liệu sử dụng trong mô hình dữ liệu quan hệ.
- SQL bao gồm cả ngôn ngữ định nghĩa dữ liệu (DDL) và ngôn ngữ thao tác dữ liệu (DML)
- MySQL là một hệ quản trị CSDL theo mô hình dữ liệu quan hệ, sử dụng SQL là ngôn ngữ dữ liệu.

Các ký pháp trong câu lệnh SQL

Ký pháp	Mô tả
VIẾT HOA	từ khoá cần thiết cho câu lệnh SQL
Viết nghiêng	Một tham số do người dùng cung cấp- thường là cần thiết
{a | b | ... }	Một tham số bắt buộc, sử dụng một trong số danh sách lựa chọn
[...]	Một tham số tùy chọn - mọi thứ trong ngoặc vuông đều là tùy chọn
tablename	Tên của bảng
column	Tên của một thuộc tính trong bảng
data type	Một định nghĩa kiểu dữ liệu hợp lệ
constraint	Một định nghĩa ràng buộc hợp lệ
condition	Một biểu thức điều kiện hợp lệ - trả về giá trị đúng hoặc sai
columnlist	Một hoặc nhiều tên cột hoặc biểu thức được phân cách nhau bởi dấu phảy
tablelist	Một hoặc nhiều tên bảng được phân cách nhau bởi dấu phảy
conditionlist	Một hoặc nhiều biểu thức điều kiện được phân cách nhau bởi dấu phảy
expression	Một giá trị đơn (ví dụ 76 or ‘married’) hoặc một công thức (ví dụ, price-10)

Chú thích trong câu lệnh SQL

- Có 3 cách viết chú thích:

1. *# This comment continues to the end of line*
2. *-- This comment continues to the end of line*
3. */* This is an in-line comment */*

/

This is a

multiple-line comment

/

Ngôn ngữ định nghĩa dữ liệu - DDL

- DDL là ngôn ngữ cho phép người quản trị CSDL hoặc người dùng mô tả và đặt tên các thực thể, thuộc tính và các quan hệ cần thiết cho ứng dụng, cùng với những ràng buộc về bảo mật và toàn vẹn liên quan.
- Kết quả của việc thực thi/biên dịch câu lệnh DDL là một tập các bảng được lưu trong các tệp đặc biệt, được gọi là danh mục hệ thống (system catalog) (hay từ điển dữ liệu/ thư mục dữ liệu).

Khởi tạo CSDL trong MySQL

- Tạo CSDL có tên là “mydb”:

CREATE DATABASE mydb;

CREATE DATABASE IF NOT EXISTS mydb;

CREATE DATABASE mydb CHARACTER **SET** utf8 **COLLATE** utf8_general_ci;

- Xoá CSDL

DROP DATABASE IF EXISTS mydb;

DROP DATABASE xyz;

- Hiển thị danh mục các CSDL đã được khởi tạo:

SHOW DATABASES;

SHOW CREATE DATABASE mydb;

- Gọi để sử dụng CSDL đã được tạo ra:

USE mydb;

Tạo tài khoản truy cập vào CSDL

- Tạo tài khoản và cho phép tài khoản này truy cập từ đâu

CREATE USER 'user'@'localhost' IDENTIFIED BY 'some_password';

Ví dụ trên tạo tài khoản user và chỉ cho phép truy cập từ máy trạm nơi cài đặt CSDL.

CREATE USER 'user'@'**%**' IDENTIFIED BY 'some_password';

Ví dụ trên tạo tài khoản user cho phép truy cập từ mọi nơi, ngoại trừ từ máy trạm nơi cài đặt CSDL

Tạo tài khoản truy cập vào CSDL

- Cấp quyền cụ thể cho tài khoản đối với các bảng trong CSDL cụ thể nào đó:

GRANT SELECT, **INSERT**, **UPDATE ON** *databaseName*.* **TO** 'userName'@'localhost';

- Các quyền cụ thể bao gồm: **SELECT / INSERT / UPDATE / DELETE / CREATE / DROP**
- Cấp mọi quyền cho một tài khoản đối với mọi bảng trong mọi CSDL:

GRANT ALL ON *.* **TO** 'userName'@'localhost' **WITH GRANT OPTION**;

WITH GRANT OPTION Cho phép người dùng này cấp quyền cho các người dùng khác.

Xem các tài khoản

- **SELECT user**,host,password **FROM** mysql.**user WHERE user IN** (‘userName', ‘abc');
- **SHOW** GRANTS FOR ‘userName'@'**%**';

Tạo bảng

```
CREATE TABLE tablename (
    column1    data type    [constraint] [,
    column2    data type    [constraint] ]  [,
    PRIMARY KEY (column1 [,column2] )]  [,
    FOREIGN KEY (column1 [,column2] ) REFERENCES tablename ] [,
    CONSTRAINT constraint ] ) ;
```

- Ví dụ:

CREATE TABLE mytable (
id **int unsigned NOT NULL auto_increment**,
username **varchar**(100) **NOT NULL**,
email **varchar**(100) **NOT NULL**,
PRIMARY KEY (id));

Tạo bảng

- Tên bảng và tên thuộc tính không nên có dấu cách, và không nên trùng với các từ khoá của SQL, ví dụ: table, first,…
- Trong trường hợp muốn tên bảng hoặc tên thuộc tính có chứa dấu cách, hoặc trùng với các từ khoá thì phải sử dụng dấu nháy đơn ‘ ‘

Ví dụ:

```
CREATE TABLE `table` (
`first name` VARCHAR(30) );
```

Khi truy vấn sau này cũng cần phải thực hiện như vậy

```
SELECT `first name`
FROM `table`
WHERE `first name` LIKE 'a%';
```

Các kiểu dữ liệu

- CHAR(n) là kiểu dữ liệu dạng chuỗi có độ dài cố định gồm n ký tự.
- Kiểu ký tự này thường đi kèm khai báo CHARACTER **SET** *codetype* để xác định số ô nhớ cho thuộc tính.

Ví dụ:

Ten CHAR(7) CHARACTER **SET** utf8mb4 sẽ tạo ra biến Ten chiếm 4*7 bytes trong bộ nhớ.

country_code CHAR(2) CHARACTER **SET** ascii tạo ra biến country_code chiếm 8*2 bytes trong bộ nhớ.

Các kiểu dữ liệu

- **DATE** kiểu ngày tháng có định dạng 'YYYY-MM-DD' với dải giá trị từ '1000-01-01' đến '9999-12-31‘
- **DATETIME** có định dạng 'YYYY-MM-DD HH:MM:SS'. Dải giá trị từ '1000-01-01 00:00:00' đến '9999-12-31 23:59:59'.
- **TIMESTAMP** có kiểu số nguyên bao gồm ngày và thời gian, có dải giá trị từ '1970-01-01 00:00:01‘ UTC đến '2038-01-19 03:14:07' UTC
- **YEAR** kiểu năm có giá trị từ 1901 đến 2155
- **TIME** kiểu thời gian với định dạng 'HH:MM:SS' và dải giá trị từ '-838:59:59' đến '838:59:59'.

Các kiểu dữ liệu

- Bộ nhớ được cấp phát

Kiểu dữ liệu	MySQL version trước 5.6.4	MySQL version 5.6.4
YEAR	1 bytes	1 bytes
DATE	3 bytes	3 bytes
TIME	3 bytes	3 bytes + tỉ số phần dư của giây
DATETIME	8 bytes	5 bytes + tỉ số phần dư của giây
TIMESTAMP	4 bytes	4 bytes + tỉ số phần dư của giây

- Phần dư của giây:

tỉ số phần dư của giây	Bộ nhớ cần dùng
0	0 bytes
1, 2	1 bytes
3, 4	2 bytes
5, 6	3 bytes

Các kiểu dữ liệu

- INTs
- FLOAT, DOUBLE, DECIMAL
- Strings (CHARs, TEXT, etc)
- BINARY, BLOB (Binary Large Object)
- DATETIME, TIMESTAMP,
- ENUM and SET

Tự động chuyển kiểu dữ liệu

- Với câu lệnh: **SELECT** ‘234’ * 2;

MySQL sẽ tự động chuyển kiểu chuỗi ‘234’ sang kiểu số và trả về kết quả 468

- Trường hợp kiểu chuỗi có lẫn ký tự số với ký tự văn bản, MySQL sẽ chuyển kiểu từng ký tự một từ trái sang phải, nếu gặp ký tự không chuyển kiểu được thì sẽ trả về giá trị là 0

SELECT ‘234abc’ * 2;

Trả về kết quả 468

SELECT ‘234abc’ * 2;

Trả về kết quả 0

Các kiểu dữ liệu dạng số

Dạng dữ liệu	Các kiểu dữ liệu
Số nguyên	**INTEGER**, **INT**, **SMALLINT**, **TINYINT**, **MEDIUMINT**, **BIGINT**
Số thập phân cố định	**DECIMAL**, **NUMERIC**
Số thực	**FLOAT**, **DOUBLE**
Kiểu bit	**BIT**

Số nguyên

- Số mặc định nhỏ nhất (Unsigned) là 0

Kiểu	Bộ nhớ (bytes)	Giá trị nhỏ nhất (Signed)	Giá trị lớn nhất (Signed)	Giá trị lớn nhất (Unsigned)
TINYINT	1	-128	127	255
SMALLINT	2	-32,768	32,767	65,535
MEDIUMINT	3	-8,388,608	8,388,607	16,777,215
INT	4	-2,147,483,648	2,147,483,647	4,294,967,295
BIGINT	8	-9,223,372,036,854,775,808 (-2^{63})	9,223,372,036,854,775,807 (2^{63}-1)	18,446,744,073,709,551,615 (2^{64}-1)

Số thập phân cố định

- Khai báo: **DECIMAL**(M,N) hoặc **NUMERIC**(M,N)

Ví dụ: budget **DECIMAL**(5,2)

Khởi tạo biết budget có kiểu số thập phân có 5 chữ số với 2 chữ số thập phân. Giá trị của budget sẽ từ -999.99 đến 999.99

Nếu khai báo budget **DECIMAL**(5) thì mặc định không có phần thập phân, tương đương với **DECIMAL**(5,0).

Nếu khai báo budget **DECIMAL** thì mặc định sẽ là **DECIMAL**(10)

Số thực

- Khai báo: FLOAT và DOUBLE

Kiểu	Bộ nhớ	Độ chính xác	Dải giá trị
FLOAT	4 bytes	2^{23} với 7 chữ số thập phân	$10^{+/-38}$
DOUBLE	8 bytes	2^{53} với 16 chữ số thập phân	$10^{+/-308}$

- REAL tương đương với FLOAT
- DOUBLE PRECISION tương đương với DOUBLE
- Mặc dù có thể khai báo REAL(M,N) hoặc DOUBLE(M,N) nhưng không nên sử dụng cách này vì các giá trị có thể bị làm tròn 2 lần, hoặc có thể bị cắt ngắn làm cho không chính xác.

Kiểu BIT

- **BIT**(M) tạo ra kiểu dữ liệu dạng bit với M bit được sử dụng. M có thể mang giá trị từ 1 đến 64
- Ta cũng có thể chỉ định giá trị cụ thể kiểu bit như sau

b'111' -> 7

b'10000000' -> 128

Tạo khoá cho bảng

- Khoá chính: **PRIMARY KEY** (someKey). khoá có thể bao gồm một hoặc nhiều thuộc tính.
  - Khoá có 1 thuộc tính có thể khai báo cùng với thuộc tính
  - Khoá gồm nhiều thuộc tính bắt buộc phải khai báo sau thuộc tính
  - Các bộ giá trị của khoá không được trùng nhau.
  - Các thuộc tính khoá không được rỗng.
- **AUTO_INCREMENT INT** cũng có thể được coi là khoá. Thuộc tính này sẽ tự động tăng giá trị lên 1 khi nhập một bản ghi mới
  - Giá trị của thuộc tính không tự động set về giá trị ban đầu nếu như có các bản ghi bị xoá.

Khoá ngoại

```
CREATE TABLE Beers (
        name CHAR(20) PRIMARY KEY,
        manf CHAR(20)
);
CREATE TABLE Sells (
        bar CHAR(20),
        beer CHAR(20) REFERENCES Beers(name),
        price REAL
);
```

Hoặc:

```
CREATE TABLE Sells (
        bar CHAR(20),
        beer CHAR(20),
        price REAL,
        FOREIGN KEY beer REFERENCES Beers(name)
);
```

Quy tắc cho khoá ngoại

Thêm ON [DELETE, UPDATE] [CASCADE, SET NULL] trong quá trình khai báo khóa ngoại.

Ví dụ:

```
CREATE TABLE Sells (
        bar CHAR(20),
        beer CHAR(20),
        price REAL,
        FOREIGN KEY beer REFERENCES Beers(name)
                ON DELETE SET NULL
                ON UPDATE CASCADE
);
```

Một số đặc tính khác cho thuộc tính

1. NOT NULL = mọi bản ghi phải có một giá trị nào đó tại thuộc tính này.
2. DEFAULT *value* = đặt mặc định một giá trị cho thuộc tính này trong trường hợp không nhập giá trị cho nó.

VD:

```
CREATE TABLE Employee (
    name CHAR(30) PRIMARY KEY,
    addr CHAR(50) DEFAULT '123 Sesame St',
    phone CHAR(16)
);
```

Sao chép bảng

- **CREATE TABLE** yourTable **LIKE** myTable;

Bảng yourTable sẽ được tạo ra với cấu trúc giống hệt myTable

- **CREATE TABLE** ClonedTable **SELECT * FROM** myTable;

Bảng CloneTable được tạo ra với cấu trúc cùng dữ liệu giống hệt myTable

Sao chép bảng

- Tạo bảng mới với dữ liệu từ câu lệnh SELECT

CREATE TABLE ModifiedPersons

SELECT PersonID, FirstName + LastName **AS** FullName **FROM** Persons

WHERE LastName **IS NOT NULL**;

Với câu lệnh này, ModifiedPersons chưa có khoá chính. Cần phải khởi tạo khoá chính cho nó.

CREATE TABLE ModifiedPersons (**PRIMARY KEY** (PersonID))

SELECT PersonID, FirstName + LastName **AS** FullName **FROM** Persons

WHERE LastName **IS NOT NULL**;

Thêm thuộc tính hiển thị thời gian

```
CREATE TABLE `TestLastUpdate` (
`ID` INT NULL,
`Name` VARCHAR(50) NULL,
`Address` VARCHAR(50) NULL,
`LastUpdate` TIMESTAMP NULL DEFAULT
CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP);
```

Hiển thị cấu trúc dữ liệu trong bảng

- Hiển thị tất cả các bảng **SHOW tables**;
- Hiển thị thông tin của một bảng nào đó
- **SHOW CREATE TABLE** *databaseName.tableName*;
- **DESCRIBE** *databaseName.tableName*;

Trong trường hợp đã ở trong CSDL rồi thì chỉ cần gọi:

- **DESCRIBE** *tableName*;

Kết quả của câu lệnh này sẽ gồm các thông tin sau:

Field	Type	Null	Key	Default	Extra
fieldname	fieldvaluetype	NO/YES	keytype	defaultfieldvalue	

Thay đổi cấu trúc bảng

- Việc thay đổi cấu trúc của bảng được thực hiện bằng lệnh ALTER TABLE, kèm theo một từ khóa chỉ rõ việc thay đổi là gì.
- Có 3 từ khóa cơ bản được sử dụng: ADD, MODIFY, và DROP.
  - ADD cho phép thêm cột vào bảng.
  - MODIFY cho phép thay đổi đặc tính của bảng.
  - DROP cho phép xóa cột của bảng. Hầu hết các hệ CSDL quan hệ không cho phép xóa cột trừ khi không có giá trị nào trong cột muốn xóa vì nó liên quan mật thiết với các bảng dữ liệu khác.

Thay đổi cấu trúc bảng

- Thêm hoặc sửa cột

```
ALTER TABLE  tablename
 {ADD COLUMN | MODIFY} ( columnname  datatype
                        [ {ADD | MODIFY} columnname  datatype] );
```

- Thêm index cho bảng

ALTER TABLE TABLE_NAME **ADD INDEX** `index_name` (`column_name`);

ALTER TABLE TABLE_NAME **ADD INDEX** `index_name` (`col1`,`col2`);

Thay đổi cấu trúc bảng

- Xoá cột, hoặc xoá ràng buộc

```
ALTER TABLE  tablename
 DROP { PRIMARY KEY |
              COLUMN  columnname |
              CONSTRAINT  constraintname } ;
```

- Ví dụ

ALTER TABLE myTable **DROP PRIMARY KEY**;

ALTER TABLE myTable **MODIFY COLUMN** new_id **DECIMAL**(20,0) **NOT NULL PRIMARY KEY**;

Ví dụ về thay đổi cấu trúc bảng

- **ALTER TABLE** myTable **ADD COLUMN** myColumn **date NOT NULL**; *-- thêm cột myColumn*
- **ALTER TABLE** myTable **DROP COLUMN** myColumn; *-- xoá cột myColumn*
- **ALTER TABLE** myTable **MODIFY** myColumn **DATETIME NOT NULL**; *-- thay đổi kiểu dữ liệu của cột myColumn*
- **ALTER TABLE** myTable **CHANGE** myColumn newColumn **DATETIME NOT NULL**; *-- thay đổi kiểu dữ liệu và đổi tên cho cột*
- **ALTER TABLE** myTable **ADD COLUMN** new_id **INT NOT NULL AFTER** id_user; *-- thêm một cột mới đằng sau một cột đã có sẵn trong bảng*

Toán tử CHANGE

- Cấu trúc câu lệnh

ALTER TABLE table_name **CHANGE** column_name new_column_definition

ALTER TABLE `<table name>` **CHANGE** `<old name>` `<new name>` <**column** definition>;

- Ví dụ

CREATE TABLE users (

firstname **varchar**(20),

lastname **varchar**(20),

age char(2));

ALTER TABLE users **CHANGE** age **tinyint UNSIGNED NOT NULL**; -- *chuyển kiểu dữ liệu từ char sang int*

Chỉ mục (index)

- **CREATE INDEX** idx_name **ON** myTable(name); -- *tạo một chỉ mục cho cột 'name' trong bảng 'myTable'*
- Tạo chỉ mục duy nhất (UNIQUE)

CREATE UNIQUE INDEX idx_name **ON** my_table(name); -- tạo một chỉ mục duy nhất cho cột ‘name’ trong bảng ‘myTable’, tức là các giá trị trong đó không trùng nhau, nhưng có thể rỗng (NULL)

- Xoá chỉ mục

DROP INDEX idx_name **ON** my_table;

Xoá bảng

- **DROP TABLE** *tablename*;
- **DROP TABLE** myDatabase.table_name
- **DROP TABLE IF EXISTS** *tablename*;