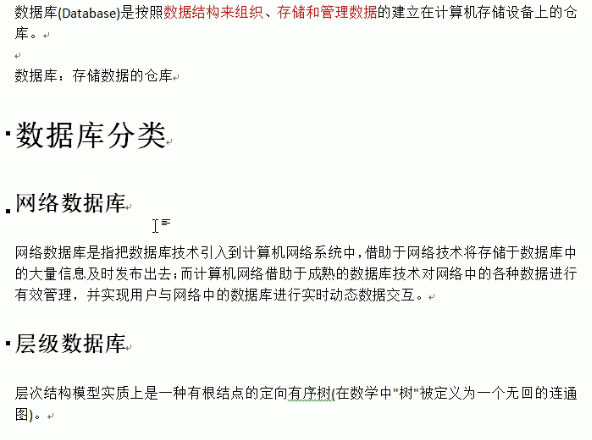
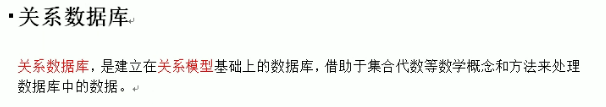
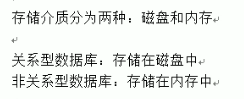
Database

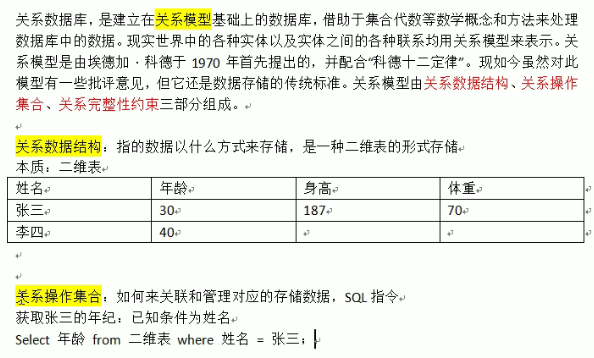


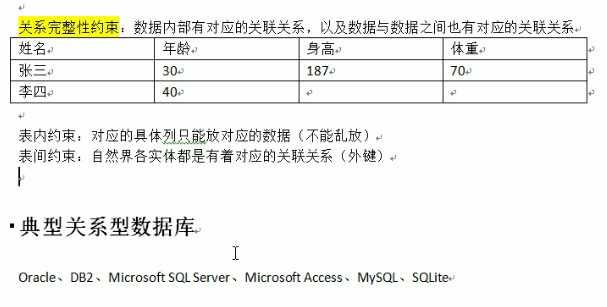




關系型數據庫,非關系型數據庫

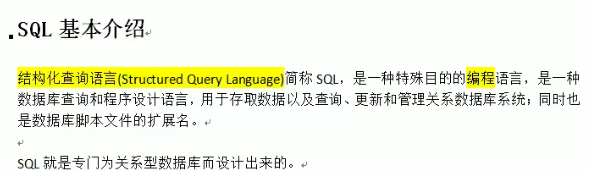
關系數據結構: 行,列的二維表形式存儲。







SQL 結構化查詢語言Structured Query Language

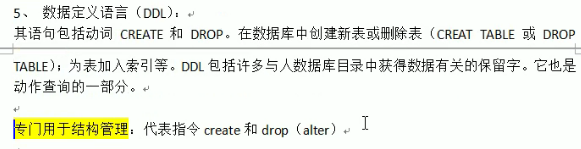


SQL分類

DQL DML TPL DCL DDL







MySQL基本介紹

MySQL是關系型數據庫管理系統。

mysql對php的支持是最好的。 wamp與lamp(window apache php mysql)

啟動與停止mysql服務

mysql是一種C/S架構: 容戶端client與服務端server

服務端對應的軟件: mysqld.exe

\*\*\*命令行方式開啟mysql

在cmd下\*\*\*\*\*

net start mysql (执行mysqld –initialize命令，，执行初始化的命令会创建默认的缺失的data目录，然后稍等片刻，再次启动即可成功)

net stop mysql

路徑: C:\Program Files\MySQL\MySQL Server 8.0\data (是由—initialize生成的)

net start mysql80 (因為mysql8.0用不同名字的exe)

net stop mysql80

路徑: C:\ProgramData\MySQL\MySQL Server 8.0\Data

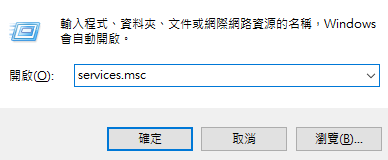
估計因為我們安裝時default是安到programdata中。

\*\*\*系統服務方式開啟mysql

前提:在安裝mysql時將mysql添加到window服務中去了。

win+R 執行 services.msc

或 我的電腦右鍵>管理>服務



開啟mysql或mysql80。

my.ini位置:

C:\ProgramData\MySQL\MySQL Server 8.0

登錄與退出mysql

通過客戶端(mysql.exe)與服務器進行連接認證,就可以進行操作。

通常: 服務端與客戶端不在同一台電腦上。

登錄:

1. 找到命令mysql.exe (通過cmd控制台, 如果有環境變量,就可以在default路徑下執行,否則要到mysql.exe所在路徑)
2. 輸入對應服務器地址: –h:host -h[ip地址/域名]
3. 輸入服務器中Mysql監聽的端口port: -P[port] mysql是3306port
4. 輸入用戶名: -u[username] –u:root
5. 輸入密碼: -p 密碼在回車之後再輸入(我們的密碼是root)

全句: mysql.exe -h:localhost –P3306 –uroot –p

連接認證基本語法:

mysql.exe/mysql –h主機地址 –P端口 –u用戶名 –p

\*\*\*\*\* 我們的mysql用戶名在mysql80 所以我們要用net start mysql80才能登錄用戶root。 \*\*\*\*\*

\*\*\*

新增mysql用戶

CREATE USER 'test'@'localhost' IDENTIFIED BY 'newpassword';

2021/10/25我們增加了一個用戶test 密碼是0000。 在net start mysql80內。

\*\*\*\*\*

1. 啟動apache httpd.exe
2. 啟動mysql服務 net start mysql80
3. 登錄mysql連接認證: mysql.exe -hlocalhost –P3306 –uroot –p (密碼root)

-P端口號通常可以不寫, 因為默認3306。

4. 退出mysql連接認證: \q 或 exit;或quit;。

通常mysql提供的服務器數量有限,建議客戶端用完,就應該斷開連接。

MySQL服務端架構

MySQL服務端架構有分以下幾層

1. 數據庫管理系統(最外層): DBMS (database management system)

專門管理服務器端的所有內容

1. 數據庫(第二層):DB , 專門用於存儲數據的倉庫(可以有多個)
2. 二維數據表(第三層): Table, 專門用於存儲具體實體的數據
3. 字段(第四層): Field, 具體存儲某種類型的數據(實際存儲單元)

數據庫常用的關鍵字:

row 行

column列(Field)

1.數據庫基本操作

基本語法: create databse 數據庫名 [庫選項];



庫選項: 數據庫相關屬性

字符集: charset, 代表當前數據庫下所有表存儲的數據默認指定的字符集(如果當前不指定, 那麼采用DBMS默認的)

校對集: collate

eg: create dabase 數據庫名 charset 字符集名;

2.顯示數據庫

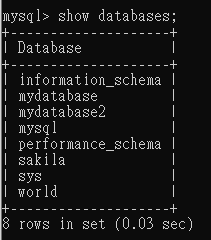
每當用戶通過SQL指令創建一個數據庫,系統就會產生一個對應的存儲數據文件夾(data), 路徑如下:



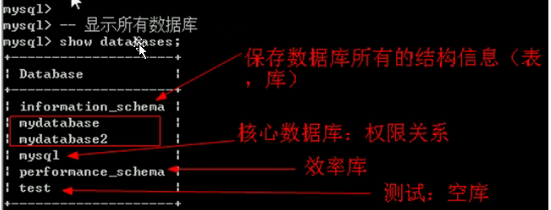
顯示全部數據庫

語法:

show databases;



* 1. version mysql 數據庫說明mysql , performance\_schema , test , information\_schema



顯示部分數據庫

show databases like ‘匹配模式’;

\_ : 匹配當前位置單個字符 (\_表示單個任何字符串)

%: 匹配指定位置多個字符 (%表示0到多個任何字符串)

匹配模式說明例子:

eg. 獲取以my開頭的所有數據庫: ‘my%’

獲取以m開頭,後面第一個字母不確定, 最後為database的數據庫: ‘m\_database’

獲取以databse為結尾的數據庫: ‘%database’;

說明: 比如我想找mydatabase2

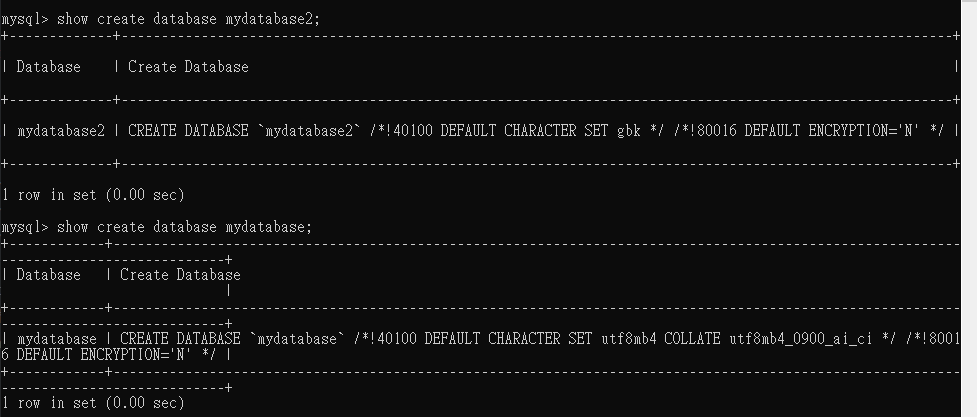
我們可以: 'm\_d\_t\_b\_s\_2';

也可以 : '%2'; , 'myda%'; , '\_yda%'; '\_yd%e%';

小心word檔的’ 號和cmd 的’號是不同的,不要直接copy上去cmd。

顯示數據庫創建語句

show create database 數據庫名;



選擇數據庫語句:

use 數據庫名;

修改數據庫的庫選項 (一旦修改成功 opt文件就會改變)

修改數據庫的字符集,校對集(庫選項)

mysql 5.5之前是可以修改數據庫名字的 rename命令。

語句:

alter database 數據庫名 charset [=] 字符集;

刪除數據庫

drop database 數據庫名;

\*\*\*

數據表操作

創建數據表

普通創建表

語法:

create table 表名(字段名 字段類型 [字段屬性] , 字段名 字段類型 [字段屬性]…. ) [表選項]; \*\*\*[表選項]是放括號外的

字段類型:

name varchar(10) --10個字符(不能超過)

將數據表創建到指定數據庫下\*\*\*\*\*

create table 數據庫名 . 表名(字段名 字段類型 [字段屬性] , 字段名 字段類型 [字段屬性]….[表選項]);

用 . 連接。

create table mydatabase.class(name varchar(10));

或 在創建數據表之前,進入數據庫(use 數據庫名)

create table class(name varchar(10));

這里name是字段名, varchar(10)是字段類型。

表選項:

Engine: 存儲引擎, mysql提供的具體存儲數據的方式, 默認有一個innodb (5.5以前默認是myisam)

charset: 字符集 (只對當前自己表有效 (級別比數據庫高))

collate: 校對集

--使用表選項:

create table student(name varchar(10)) charset utf8;

--複製已有的表結構

\*\*只會複製結構, 如果表中的數據不會複製。

語法:

create table 新表名 like 表名;

--只要使用 數據庫.表名, 就可以在任何數據庫下訪問其他數據庫的表名

--在test數據庫下創建一個與teacher一樣的表

ues test;

create table teacher like mydatabase.teacher;

--查看當前數據庫下的所有表

show tables;

frm結構文件 (我們mysql8.0是idb)



**mysql5.6 之frm,MYD,MYI,idb,文件說明**

如數據庫a，數據庫表b  
**如果表格b採用MyISAM**，data\a中會產生3個文件：  
b.frm ：描述表結構文件，字段長度等  
b.MYD(MYData)：數據信息文件，存儲數據信息(如果採用獨立表存儲模式)  
b.MYI(MYIndex)：索引信息文件。

**如果表格b採用InnoDB**，data\a中會產生1個或者2個文件：  
b.frm ：描述表結構文件，字段長度等  
如果採用獨立表存儲模式，data\a中還會產生b.idb文件（存儲數據信息和索引信息）

innoDB與MyISAM是存儲引擎。

innoDB存儲引擎所有的數據都存儲在外部的ibdata里。

(在路徑C:\ProgramData\MySQL\MySQL Server 8.0\Data)



ibdata1存儲所有innodb存儲引擎對應的表數據。

--查看當前數據庫下的所有表

show tables;

--查看匹配數據表

show tables like ‘匹配模式’;

--查看/顯示表結構

意義:顯示表中所包含的字段信息(名字,類型,屬性等)

語法:

Describe 表名

Desc 表名

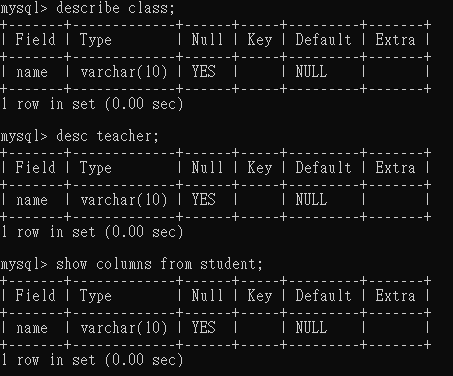
show column from 表名

三種都可以查看表結構。

describe class;

desc teacher;

show columns from student;



Field 是字段名 name

Type是字段類型 varchar(10)

Null是 值是否允許為空。

Key 是索引

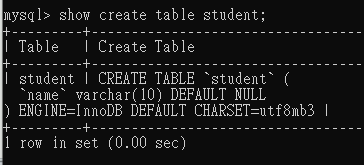
default是 值的默認值, NULL就是: 如果不給值,值就是空(Null)

Extra 是額外的屬性。

--顯示/查看表創建語句

show create table 表名;

show create table student;



* MySQL裡，utf8的是隱式的，等價於utf8mb3，但是但從名稱上是看不出最大字符數的。

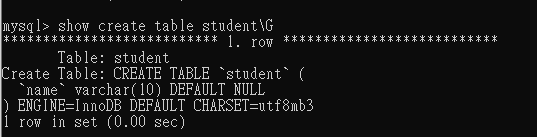
好消息是MySQL在未來會改正uft8的命名：“在未來的MySQL版本中會刪除utf8mb3，請改用utf8mb4。雖然utf8目前是utf8mb3的別名，但在未來某個時候，utf8將指代utf8mb4。為了避免對utf8的含義產生歧義，可以考慮直接把字符集引用指定為utf8mb4，而不是utf8。”

mysql中有多種結束符

;與\g 是一樣的。

\G 字段在左側直放,數據在右側橫放

show create table student\G



設置表屬性

表屬性: 指的是表選項,engine,charset,collate

--修改表屬性

語法:

alter table 表名 表選項 [=] 值;

alter table student charset = gbk;

注意:如果數據庫已確定了, 里面有很多數據,就不要較易修改表選項(字符集影響不大)。

--修改表結構

修改表名:

rename table 舊表名 to 新表名;

*--數據表名字通常有前綴, 取數據庫名的前兩個字母加上下划線*

*--我們在mydatabase庫中 所以新命名是*

rename table student to my\_student;

修改表選項:

alter table表名 表選項 [=] 新值;

alter table 表名 表選項 [=] 值;

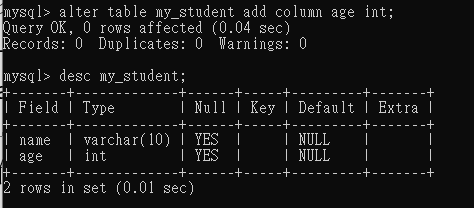
alter table student charset = gbk;

新增字段:

alter table 表名 add [column] 新字段名 列類型 [列屬性] [位置 first/after字段名];

--給student增加 age字段

alter table my\_student add column age int;



特點: 新增字段默認加到表的最後面。

字段位置:

first: 在某某之前(最前面), 第一個字段

after 字段名: 放在某個字段名後面。(默認的)

--增加字段: 放到第一個字段位

alter table my\_student add id int first;

int是列類型, first是位置(取前面)。

修改字段名:

alter table 表名 change 舊字段名 新字段名 字段類型 [列屬性] [新位置];

alter table my\_student change age ages int;

修改字段類型(屬性):

alter table 表名 modify 字段名 新類型 [新屬性] [新位置];

alter table my\_student modify name varchar(15) after id;

刪除字段:

alter table 表名 drop 字段名

alter table my\_student drop ages;

刪除表結構:

drop table 表名 [,表名2]; 可以同時刪多個數據表(數據庫不可以)。

status; 查看當前所在數據庫。

select database(); 查看當前所在數據庫。

show tables;查看當前所在數據庫與所有數據。

數據操作 數據基本操作:

插入操作:

將數據以SQL的形式存儲到指定的數據表(字段)里面。

語法:

1.insert into 表名 (字段名列表) values(對應字段列表);

注意: 後面(values中)對應的值列表 只需要與前面的字段列表相對應即可(順序不一定要與表結構一樣)。

如果結構中允許值是空的, 字段列表與對應的值不寫也可以。

2.insert into表名 values(對應表結構);

這種寫法values順序要與表結構一樣。

查詢操作：

獲取所有數據:

select \* from 表名;

select \* from my\_teacher;

查詢表中部分字段:

select 字段列表 from 表名; 字段列表用 , 號隔開

select name from my\_teacher;

select name,age from my\_teacher;

簡單條件查詢數據:

select 字段列表/\* from 表名 where 字段名 = 值;

--MySQL中 沒有==符號

*--獲取年齡為30的人名*

select \* from my\_teacher where age = 30;

select name from my\_teacher where age = 30;

刪除數據操作:

delete from 表名 [where 條件]; --如果沒有where條件, 系統會自動刪除該表中所有數據(\*\*\*慎用)

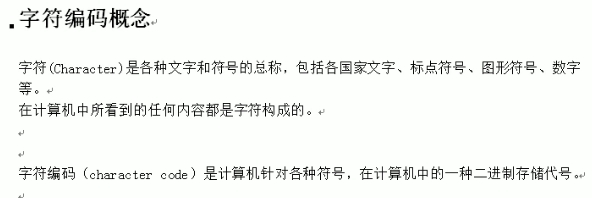
delete from my\_teacher where age >= 40;

更新(修改)數據操作: (通常是修改部分字段數據)

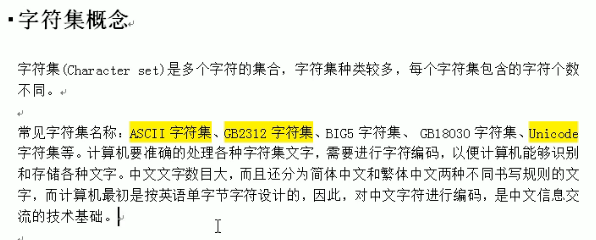
update 表名 set 字段名 = 新值 [where 條件]; --如果沒有where條件,所有表中的對應字段名都會被改成統一值。

MySQL字符集

字符編碼的概念



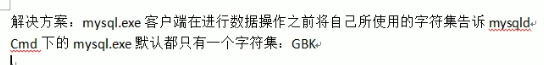
字符集的概念



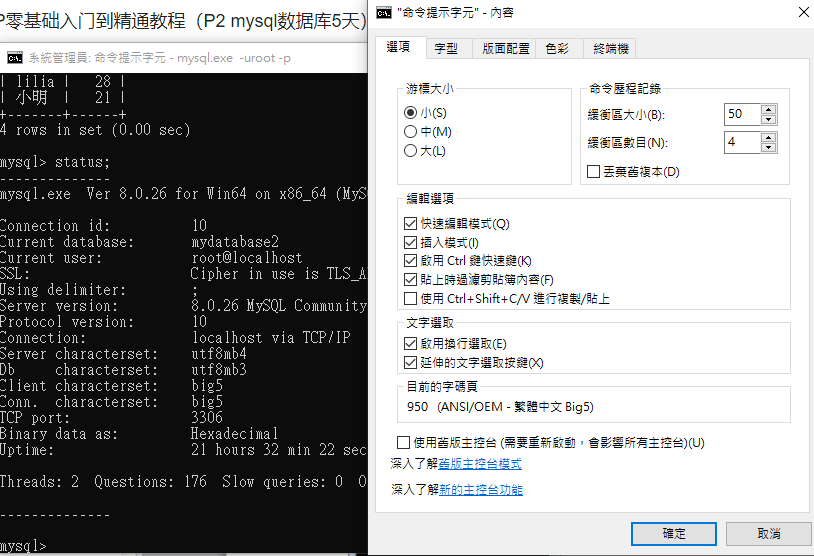
設置客戶端所有字符集







我們的電腦是默認big5。(client characterset: big5)



我們要解決的是:

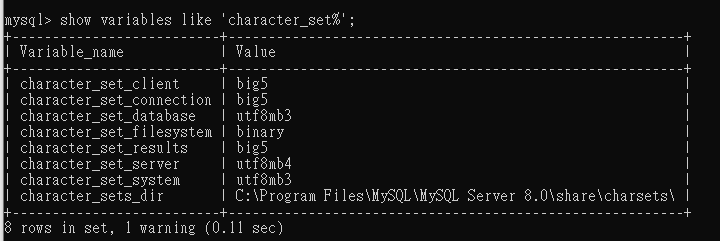
mysql.exe如何告知mysqld.exe對應的字符集類型。

可以通過:

set names 字符集;

深層原理: 客戶端,服務端, 連接層(show variables like ‘character\_set%’)

show variables like 'character\_set%';



原理: \*\*\*\*\*

mysql.exe與mysqld.exe之間的處理關系分為3層

客戶端傳入數據給服務端: client: character\_set\_client

服務端返回數據給客戶端: server: character\_set\_results

客戶端與服務端之間的連接: connection: character\_set\_connection

set names 字符集: 就是一次性打通三層關系的字符集,使之變得一致。

我們用set names big5;

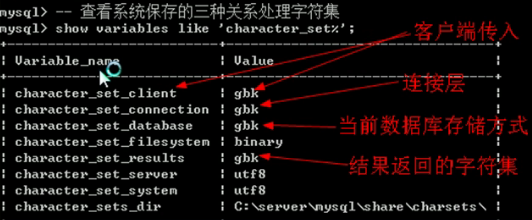
會使:

character\_set\_client

character\_set\_connection

character\_set\_results

三個變成big5。 (圖是教程,他用gbk簡體字)



修改服務器端變量的值

set 變量名 = 值;

set character\_set\_client = utf8;

set character\_set\_results = utf8;

connection 只是為了更方便客戶端與服務端進行字符集轉換而設,不改也可以。

set names 字符集;

set names 字符集可以直接改3個,results,connection,client;

字段類型(列類型)

整型(整數類型) int

tinyint (迷你整形): 系統用1個字節來保存整形: 1個字節=8位, 最大能表示的數值是0~255。

smallint(小整形): 系統用2個字節來保存整形: 最大能表示的數值是0~65535。

mediumint(中整形): 系統用3個字節來保存整形。

int(整形): 標準整形, 系統用4個字節來保存整形。

bigint(大整形): 系統用8個字節來保存整形。

tinyint是能放255, 但有分正負數, range是 -128到127。



無符號標識設定

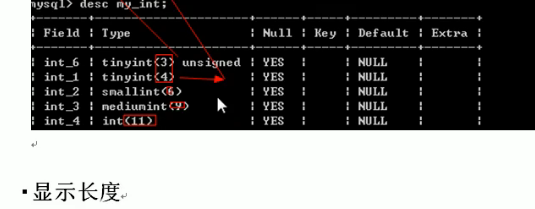
無符號: 表示存儲的數據在當前字段中, 沒有負數(只有正數,tinyint就能0-255)

語法:

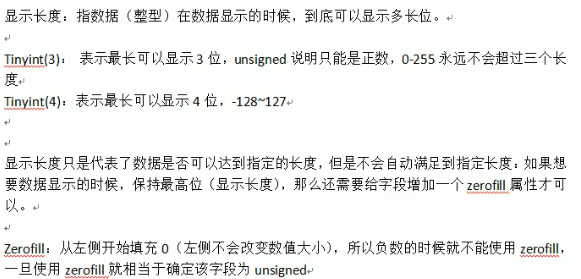
在類型之後加上unsigned

alter table my\_int add int\_6 tinyint unsigned first;

顯示長度 (舊版mysql才有)



zerofill: 一旦使用zerofill 字段就為unsigned (無負數)。

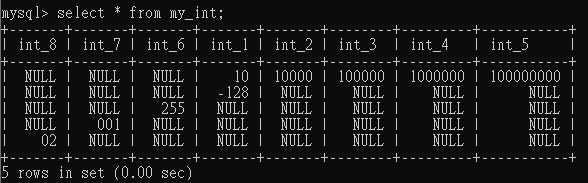




指定zerofill長度:

alter table my\_int add int\_8 tinyint(2) zerofill first;

insert into my\_int (int\_8) value (2);





小數類型:

在mysql中, 將小數類型分為2種: 浮點型,定點型。

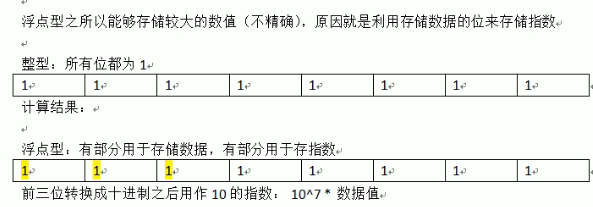
浮點型:

浮點型又稱精度類型: 是一種有可能丟失精度的類型。

float 單精度類型: 系統提供4個字節用來存儲數據, 但是能表示的數據範圍比整數類型大得多, 大概是10^38, 但只能保證7個左右的精度(如果數據在7位數以內,基本是精確的, 如果超出7位數,就不精確了)

double 雙精度類型: 系統提供8個字節用來存儲數據, 大概是10^308,但精度只有15位左右。

浮點型之所以能夠存儲較大的數值(不精確),原因是利用存儲數據的位來存儲指數。



語法:

float:表示不指定小數位的浮點數

float(M,D): 表示一共存儲M個有效數字, 其中小數部分占D位。

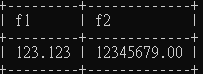
例:

float(10,2): 整數部分8位,小數部分2位。

*--創建float表*

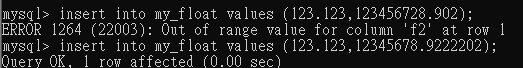
create table my\_float (f1 float , f2 float(10,2)) charset = utf8;

insert into my\_float values (123.123,12345678.90);



看出精度不夠,出錯了 12345679.00。

如果精度丟失,浮點型是按四舍五入的方式進行計算。



\*\*\*整數部分不能超出範圍,但小數部分可以。

浮點數可以用數學計數法存儲數據

浮點數的應用



定點數:

定點數:

能夠保證數據精確的小數(超出長度會四舍五入),整數部分一定精確。

Decimal

Decimal定點數:

系統自動根源存儲的數據來分配存儲空間, 每大概9個數就會分4個字節來進行存儲,同時小數和整數部分是分開的。

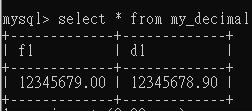
語法:

decimal(M,D): M表示總長度,最大值不能超過65. D表示小數部分,最大不能超過30。

*--創建decimal表*

create table my\_decimal(f1 float(10,2),d1 decimal(10,2)) charset = utf8;

insert into my\_decimal values (12345678.90,12345678.90);



可以看出 decimal定點數的精度是較高的。

插入最大數

insert into my\_decimal values (99999999.99,99999999.99);

插入最大數(報錯)

insert into my\_decimal values (99999999.99,99999999.999);

定點數如果整數部分進位後超出長度也會報錯。

定點數的應用



時間日期型

date time datetime timestamp year

Date:

日期類型, 系統使用3個字節來存儲數據, 格式為 YYYY-mm-dd 能表示的範圍是

由1000-01-01至9999-12-12, 初始值為0000-00-00

Time:

時間類型, 能夠表示某個指定時間,但是系統同樣是3個字節存儲,

格式為HH:ii:ss , 但是mysql中time類型能夠表示的時間範圍要大得多,

能表示 -838:59:59 ~ 838:59:59 在mysql中具體的用處是用來描述時間段。

Datetime:

日期時間類型, 就是date和time的結合.使用8個字節存儲。

格式為YYYY-mm-dd HH:ii:ss , 能表示的

區間1000-01-01 00:00:00到9999-12-31 23:59:59,

也可以為0值:

0000-00-00 00:00:00

Timestamp:

時間戳類型: mysql中的時間戮只是表示 由格林威治事件開始,

但還格式是 YYYY-mm-dd HH:ii:ss

Year:

年類型: 使用1個字節存儲. 只能夠表示1900~2155年之間。

但是year有兩種數據插入方式: 0~99和四位數的具體年

--創建時間日期類型表

use mydatabase2;

create table my\_datetime (

    d1 date ,

    d2 time ,

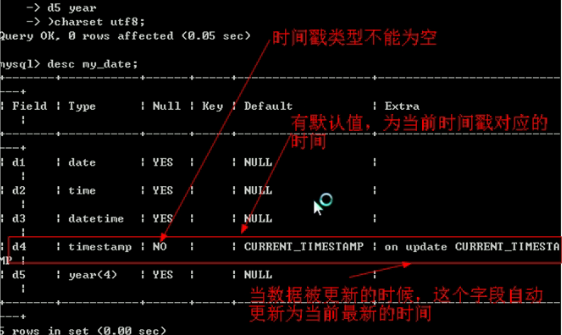
    d3 datetime ,

    d4 timestamp ,

    d5 year

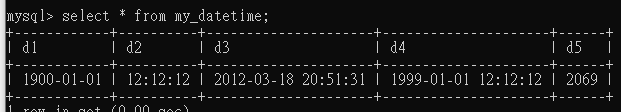
    ) charset = utf8;

mysql 8.0的timestamp值允許為空。 舊版的timestamp



插入時間數據:

insert into my\_datetime values ('1900-01-01','12:12:12','2012-03-18 20:51:31','1999-01-01 12:12:12',69);



year可以用4位數插入,2位數插入也可以。

\*\*

year如果插入2位數時,

69或以下是2000年+69以下的數。

70或以上是1900件+70或以上的數。

timestamp當對應的數據被修改的時候, 會自動更新(這個被修改的數據不是自己)

update my\_datetime set d1 = '2000-01-01' where d5 = 2069;

mysql8.0 timestamp不會自動改了。

time類型特殊性: 本質是用來表示時間區間, 能表示範圍較大。

insert into my\_datetime values  ('1900-01-01','512:12:12','2012-03-18 20:51:31','1999-01-01 12:12:12',2015);

在進行時間類型錄入時(time) , 還可以使用一個簡單日期代替時間, 在時間格式之前加一個空格, 之後指定數字: 系統會自動將數字轉換成天數 \* 24小時, 再加上後面的時間。

insert into my\_datetime values  ('1900-01-01','3 12:12:12','2012-03-18 20:51:31','1999-01-01 12:12:12',2000);

php中有強大的時間日期轉換函數, date將時間戳轉換成想要的格式: strtotime又可以將行多格式轉換成對應的時間戳。

php通常不需要數據庫來幫助處理這麼複雜的時間日期。

所以通常配合php的時候, 時間的保存通常使用真正的時間戳。從而用整型來保存。

字符串型 char varchar text enum

char

定長字符: 指定長度之後,系統一定會分配指定的空間用於存儲數據。

char(L) , L代表字符數(中文與英文一樣) , L長度成0~255。

varchar

變長字符: 指定長度之後,系統會根據實際存儲的數據來計算長度,分配合適的長度(數據沒有超出長度)

varchar(L) , L代表字符數(中文與英文一樣) , L長度成0~65535。

因為varchar要記錄數據長度(系統會根據數據長度分配空間), 所以每個varchar數據產生後, 系統會在數據後面增加1~2個字節的額外空間開銷。

如果數據本身小於127個字符, 就額外開銷1個字節,

如果數據本身大於127個字符, 就額外開銷2個字節。



char 與varchar 區別\*\*\*

1. char會使用指定空間, varchar是根據數據來定空間。
2. char的數據查詢效率比varchar高。
3. 如果確定數據是一定占指定長度。那麼使用char類型,
4. 如果不確定數據到底是多小長度,那麼使用varchar類型。
5. 如果數據長過255字符,不論是否固定長度,都使用text。

text

文本類型 本質上mysql提供了2種文本類型。

text: 存儲普通的字符文本。

blob: 存儲二進制文本(圖片,文件) 一般不會用blob來存儲文件本身,通常是使用一個鏈接來指向對應的文件本身。

text: 系統中提供了四種text

tinytext: 系統使用1個字節來保存, 實際能夠存儲數據為: 2 ^ 8 +1

text: 系統使用2個字節來保存, 實際能夠存儲數據為: 2 ^ 16 +2

mediumtext:系統使用3個字節來保存, 實際能夠存儲數據為: 2 ^ 24 +3

longtext:系統使用4個字節來保存, 實際能夠存儲數據為: 2 ^ 32 +4

\*\*\*\*\*

注意: 在選擇對應的存儲文本時, 不用刻意去選擇text類型, 系統會自動根據存儲的數據長度來自動選擇合適的類型。

注意: 在選擇字符存儲的時候, 如果數據超過255個字符, 那麼一定選擇text存儲。

enum

枚舉類型: 在數據插入之前, 先設定好幾個值, 這幾個值就是最終可以出現的結果。

如果確定某數據只有幾個值: 比如: 男,女,保密。

就可以用enum枚舉類型。

語法:

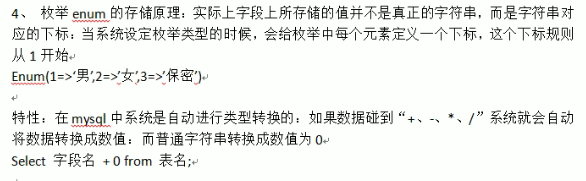
enum(值1,值2…);

系統提供了1~2個字節來存儲枚舉類型, 通過計算enum列舉的具體值來選擇實際的存儲空間, 如果數據值列表在255個以內, 那麼一個字節就夠了。

如果256~65535個數據值列表, 就2個字節。

create table my\_enum (gender enum('男','女','保密')) charset = utf8;

枚舉類型 enum 存儲原理



enum 索引下標

select gender + 0 from my\_enum;

enum插入方法2

enum可以 以下標索引的方式插入數據。

注意: enum的下標是由1開始, 不是由0開始。

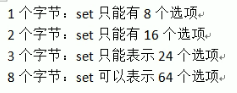
set

集合: 是一種將多個數據選項可以同時保存的數據類型, 本質是將指定的項按照對應的二進制位來進行控制, 1表示該選項被選中, 2表示該選項沒有被選中。

語法:

set(‘值1’, ‘值2’,…);

系統為set提供多個字節進行存儲,但是系統會自動計算來選擇存儲單元。



set和enum一樣, 最終存儲到數據字段中的是數字(下標索引)而不是真實的字符串。

create table my\_set (hobby set('reading','workout','table tenis','watch ball')) charset = utf8;

create table my\_set2 (interest set('書','健身','乒乓球','看球')) charset = utf8;

create table my\_set2 (interest set('看書','健身','乒乓球','看球')) charset = big5;

ERROR 3507 (HY000): Failed to update columns dictionary object.

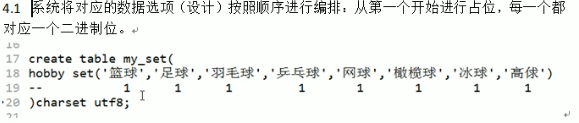
set中文選項會報錯. 用big5就不報錯, 可能因為我們的cmd是big5。

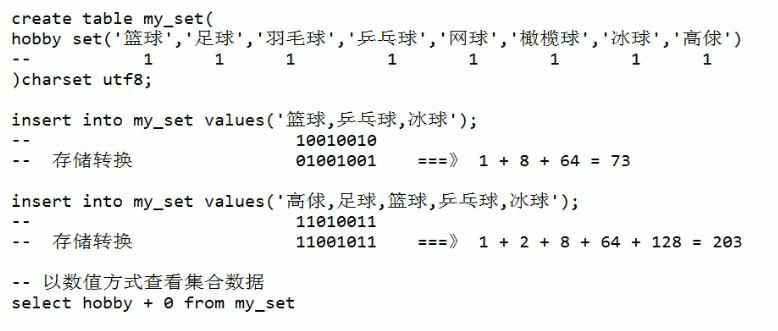
插入數據  
可以插入多個數據, 就是在數據插入的字符串中, 使用對應的 ,號 分開。將選項進行隔開。

insert into my\_set values('reading,table tenis');

insert into my\_set values('workout,table tenis,reading');

注意: , 號中間不可以有空格。





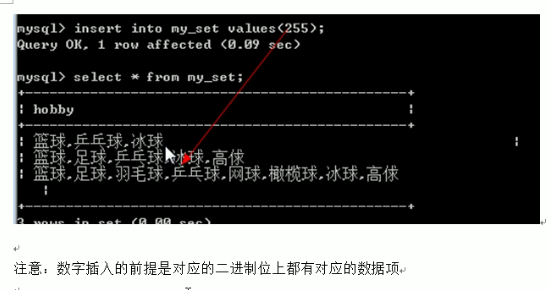
以數值方式查看集合數據

select hobby + 0 from my\_set;

insert into my\_set values(255);

1個字節16個選項 > 1個字節255

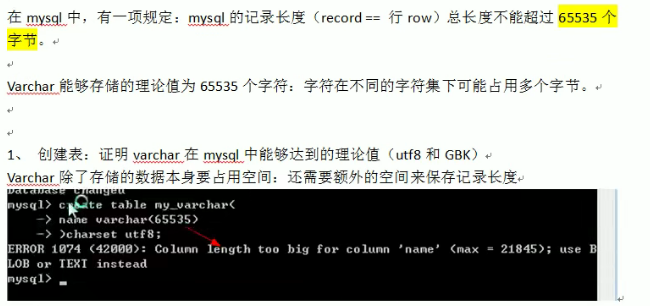
2個字節…個選項



enum 單選

set 多選

mysql記錄長度 (varchar最大長度)





\*\*\*

字段屬性 列屬性

在mysql中一共有6種屬性:

null, 默認值, 列描述 , 主鍵, 唯一鍵, 自動增長。

null屬性

NULL代表字段為空。NULL默認YES

如果要默認NULL是NO,

語法 NOT NULL

create table my\_default (name varchar(10) NOT NULL) charset = utf8;





默認值

Default: 默認值,當字段被設計時, 如果允許默認條件下, 用戶不進行數據插入,那就可以使用事先默認的數據來填充, 通常default為NULL;

語法: default 值 (黃色部分)

create table my\_default (name varchar(10) NOT NULL ,

age int default 18) charset = utf8;

insert into my\_default (name) values ('peter');

2.insert時告知mysql使用默認值。

insert into my\_default values ('jack',default);

列描述:

comment, 是專門用於給開發人員進行維護的一個注釋說明。

語法:

comment ‘字段描述’;

create table my\_comment (

    name varchar(10) not null comment '當前是用戶名,不能為空值',

    pass varchar(50) not null comment '密碼不能為空'

) charset = utf8;

查看comment: 必須通過查看 表創建語句

show create table my\_comment;

主鍵

主要的鍵 primary key, 在一張表中的字段 , 里面的值具有唯一性。

創建主鍵:

隨表創建

系統提供兩種增加主鍵的方式

1. 直接在需要當做主鍵的字段後, 增加 primary key屬性來確定主鍵。

create table my\_pri1 (username varchar(10) primary key )charset = big5;

1. 在所有字段創建之後增加primary key選項, primary key(字段信息);

create table my\_pri2(username varchar(10) , primary key(username)) charset = big5;

表後創建(表而創建後 增加primary key 屬性)

alter table 表名 add primary key(字段名);

create table my\_pri3 (username varchar(10)) charset = big5;

alter table my\_pri3 add primary key(username);

查看主鍵

主鍵不允許為空: 如果給了primary key, Null屬性就會自動not null;

desc 表名;

show create table 表名;

刪除主鍵

alter table 表名 drop primary key;

但null 不會自動改回yes;

alter table my\_pri3 modify username varchar(10) NULL;

複合主鍵 (複合主鍵時,單個主鍵沒有唯一性,全部複合主鍵有唯一性)

create table my\_score(

    student\_no char(10),

    course\_no char(10),

    score tinyint not null,

    primary key(student\_no , course\_no)) charset big5;

學號與課程都有唯一性。

insert into my\_score values ('0001','course001',100);

insert into my\_score values ('0001','course002',90);

insert into my\_score values ('0002','course001',65);

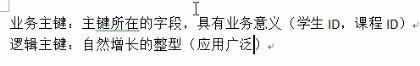
主鍵約束:

主鍵的字段的數據不能有重複,

主鍵的字段的數據不能為空。

主鍵分類:



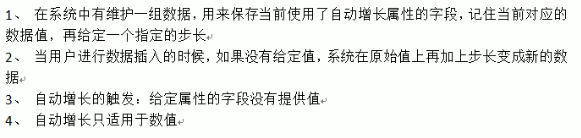


自動增長

auot\_increment 當給定某個字段該屬性之後, 該列的數據在沒有提供確定數據時,系統會根據之前已經存在的數據進行自動增加, 之後填充數據。

通常用於邏輯主鍵。

原理: (自動增長只用於數值)



語法:

在字段後 增加 auto\_increment屬性

create table my\_auto(

    id int primary key auto\_increment,

    name varchar(10) not null comment '用戶名',

    pass varchar(10) not null comment '密碼'

)charset = big5;

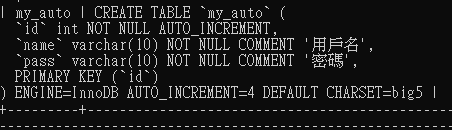
插入數據,觸發自動增長。

insert into my\_auto values(null,'tom','123456');

我們給auto\_increment的字段的值為null

查看自動增長\*\*\*

show create table my\_auto; 會有一個auto\_increment的值,說明,一張表只能有一個字段是auto\_increment。



修改自動增長的值

alter table 表名 auto\_increment = 值;

刪除自動增長

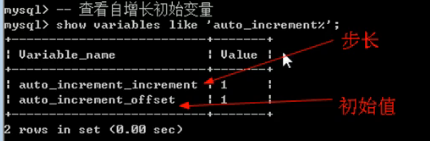
在自動增長所在的字段, 改為沒有自動增長auto\_increment

alter table my\_auto modify id int; (不要再增加primary key)

初始設置

查看自增長初始值

show variables like ‘auto\_increment%’;



auto\_increment的值不能向下修改

auto\_increment插入值時,可以是未用而且小於當前auto\_increment的值。

但插入了未用而且小於當前的值之後,auto\_increment不會改變。

唯一鍵

unique key, 用來保證對應的字段中的數據唯一的。

主鍵也可以用來保證字段數據唯一性,但是一張表只有一個主鍵/複合主鍵。

1. unique key一張表可以有多個
2. unique key允許字段中數據為NULL,NULL可以有多個(NULL不參與比較)

創建unique key

語法:

1. 直接在表字段後增加 unique

create table my\_unique1(

    id int primary key auto\_increment,

    username varchar(10) unique

) charset = big5;

1. 在所有字段之後加上 unique key(字段名列表)

create table my\_unique2(

    id int primary key auto\_increment,

    username varchar(10),

    unique key (username)

) charset = big5;

1. 創建完表之後也可以加上unique key

alter table 表名 add unique key(字段名列表)

create table my\_unique3(

    id int primary key auto\_increment,

    username varchar(10)

) charset = big5;

alter table my\_unique3 add unique key (username);

查看唯一鍵

desc 表名;

unique key只有NULL時可以重複。

刪除唯一鍵

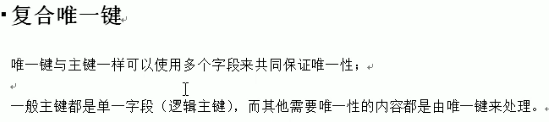
一個表中允許存在多個唯一鍵, 所以語法與主鍵不同。

alter table 表名 drop index 唯一鍵字段名;

alter table my\_unique3 drop index username;

複合唯一鍵

與主鍵primary key相同,唯一鍵也可複合。



複合唯一鍵

alter table my\_unique3 add password varchar(10) unique;

insert into my\_unique3 values(null,'hello1997','hihipass');

insert into my\_unique3 values(null,'hello1997','hihipass2'); //不能

複合唯一鍵:

\*\*\*\*\*

複合唯一鍵unique key是 || 邏輯或 不能重複。

複合主鍵primary key 是 && 邏輯與 不能重複。

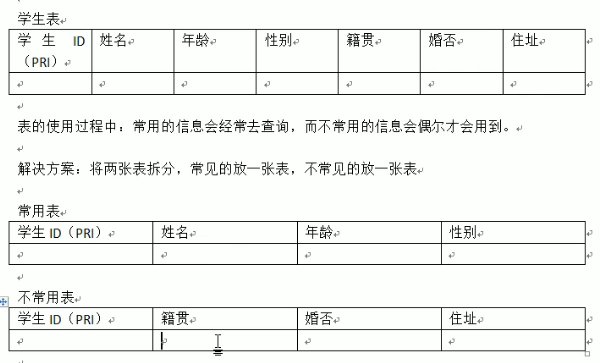
表關系

表與表之間(實體)有什麼的關系,每種關系應該如何設計表結構。

一對一\*\*\*

一張表中的一條記錄與另外一張表中最多有一條明確的關系。

通常,此設計方案保證兩張表中使用同樣的主鍵即可。



一對多 (多對一)

通常一對多的關系設計的方案, 在多關系的表中去維護一個字段,這個字段是 ‘一’關系的主鍵。



多對多

一張表中的一條記錄在另外一張表中可以匹配到多條記錄, 反過來也一樣。

多對多關系如果按照多對一的關系維護, 就會出現一個字段中有多個其他表的主鍵。在訪問時就會帶來不便。

既然通過兩張表自己增加字段解決不了問題, 那麼就用第三張表解決。

eg. 師生關系

一個老師教多個學生。

一個學生有多個老師。



多對多解決方案: 增加一張中間表,記錄對應的其他表形成的兩個多對一關系。

多對一的解決方案是在’多’表中增加’一’表對應的主鍵字段。

高級數據操作

**新增數據**

多數據插入

只寫一次insert ,但是可以插入多條記錄。

insert into 表名 [字段名列表] values (值列表),(值列表)…;

create table my\_insert(

    id int primary key auto\_increment,

    username varchar(10) not null unique,

    password varchar(10) not null

)charset = big5;

主鍵衝突

在有的表中, 使用的是業務主鍵(字段有業務含義), 但是往往在進行數據插入時, 又不確定數據表中是否已經存在對應的主鍵。

主鍵衝突的解決方案:

1. 主鍵衝突更新

類似插入數據語法, 如果插入時主鍵衝突, 那麼采用更新方法

insert into 表名 [字段名列表] values (值列表) on duplicate key update 字段=新值;

insert into my\_student values ('0003','小美') on duplicate key update stu\_name = '小美';

會直接取代主鍵衝突的原數據。

1. 主鍵衝突替換 (內存效率比上一種方法低)

當主鍵衝突之後, 幹掉原數據, 重新插入進去。

replace into [字段名列表] values (值列表);

replace into my\_student values ('0001','新人');

蠕蟲複制

一分為二,成倍增加。

由已有的數據中獲取數據, 并且將獲取到的數據插入到數據表中。

insert into 表名 [字段名列表] select \*/字段名列表 from 表名;

注意: 蠕蟲複制的確是重複數據, 沒有太大的業務意義, 可以在短期內快速增加表的數據量, 可以測試表的壓力, 還可以通過大量數據來測試表的效率(索引)。

蠕蟲複制也要注意主鍵衝突。

高級數據操作

更新數據

注意: 更新數據時如果沒有條件, 會是全表更新, 要慎用。

可以用limit 來顯示更新的數量;

update 表名 set 字段名 = 值 where 條件;

update 表名 set 字段名 = 值 [where 條件] limit 數量;

update my\_simple set name = 'e' where name = 'a' limit 4;

刪除數據

delete 不會重置auto\_increment;

即使清除整張表的內容。

delete from 表名;

重置auto\_increment

truncate 表名; 等於 drop整張表再create一張新表。(重置auto\_increment)

truncate my\_auto;

高級操作

查詢數據

完整的查詢指令

select select選項 字段名列表 from 數據源 where 條件 group by 分組 having 條件 order by 條件 limit 限制;

select選項: 系統如何對待查詢得到的結果

all: 默認的, 表示保存所有的記錄

select all \* from my\_simple;

等於 select \* from my\_simple;

distinct: 去重複, 只保留一條(所有的字段都相同時)

select distinct \* from my\_simple;

字段名列表:

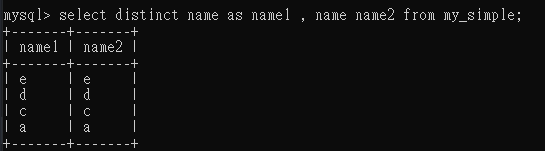
有時候需要從多張表獲取數據,在獲取數據時,可能存在不同表中有同名的字段。

這時就需要將同名的字段命名成不同名的: 別名alias。

語法:

字段名 [as] 別名;

select distinct name as name1 , name name2 from my\_simple;



from 數據源

1.單表數據

from 表名;

2.多表數據

from 表1,表2…;

會把兩張表的數據 相乘, 字段數拼接。

多表數據的本質:



3.動態數據

from 後面跟的數據不是一張實體表, 而是一個表中查詢出來得到的二維結果表(子查詢)。

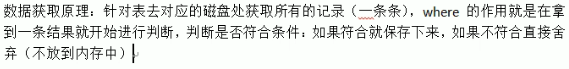
語法:

from (select 字段名列表 from表) as 別名;

where 字句

用來從數據表獲取數據時, 進行條件篩選。

原理:



group by 字句

根據指定的字段, 將數據進行分組: 分組的目標是為了統計。

分組統計:

group by 字段名;

select \* from my\_student group by class\_id;

group by是為了分組後進行統計, 如果只是想查看數據, 那麼group by沒什麼含義, group by將數據按照指定的字段分組後, 只會保留每組的第一條記錄。

利用一些統計函數: (也叫聚合函數)

count() 統計每組中的數量, 如果統計目標是字段, 那麼不統計為空NULL字段。

avg() 平均值

sum() 總和

max() 獲取最大值

min() 獲取最小值

按照班級統計每班人數, 最大age,最小height,平均年齡。

select class\_id, count(\*),max(stu\_age),min(stu\_height),avg(stu\_age) from my\_student group by class\_id;

group\_concat(): 為了將分組中指定的字段進行合并(字符串拼接)。

group\_concat();的使用  將分組中指定的字段進行合并(字符串拼接)。

select class\_id,group\_concat(stu\_name), count(\*),max(stu\_age),min(stu\_height),avg(stu\_age) from my\_student group by class\_id;

多分組

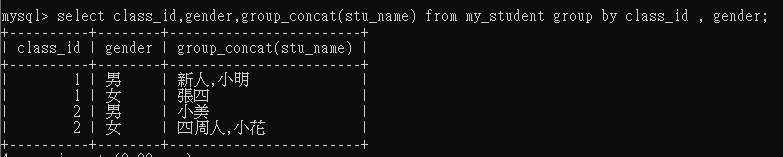
將數據按照某個字段進行分組之後, 對已經分組的數據進行再次分組。

語法:

group by 字段1, 字段2;

--先按字段1分組,再按字段2分組

select class\_id,gender,group\_concat(stu\_name) from my\_student group by class\_id , gender;



分組排序

mysql中,分組默認有排序功能, 按照分組字段進行排序, 默認是升序。

語法:

group by 字段 , 字段order by字段 [asc | desc] , 字段 [asc | desc] ;

分組排序 (class\_id升序,gender降序)

select class\_id,gender,group\_concat(stu\_name) from my\_student group by class\_id , gender order by class\_id asc , gender desc;

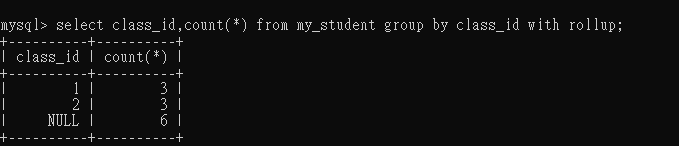
在mysql8.0中, asc 與desc好似只能在order by中使用。

回溯統計

當分組進行多分組之後, 往上統計的過程中, 需要進行層層上報,將這種層層上報統計的過程叫做回溯統計,每一次分組向上統計的過程都會產生一次新的統計數據, 而且當前數據對應的分組字段為NULL。

語法:

group by 字段 with rollup;



rollup 總計了的結果 就是NULL 6。

select class\_id,count(\*) from my\_student group by class\_id;

select class\_id,count(\*) from my\_student group by class\_id with rollup;

select class\_id,gender ,count(\*) from my\_student group by class\_id , gender;

select class\_id,gender ,count(\*) from my\_student group by class\_id , gender with rollup;

having 字句

having與where都是用來進行條件篩選。

having是在group by字句之後,可以針對分組數據進行統計篩選, 但是where不行。

查詢班級人數大於4個的。

where不能使用聚合函數: 聚合函數是用在group by分組的時候 where已經運行完。

只能用having, having在group by分組之後, 可使用聚合函數或者字段別名(where是從表中取出數據, 別名是在數據進入到內存之後才有的)

select class\_id,count(\*) as number from my\_student group by class\_id having count(\*) >=4 ;

強調: having是在group by之後, group by是在where之後

where的時候表示將數據從磁盤拿到內存, where之後所有操作都是內存操作。

order by 字句

order by排序; 根據規則進行排序。

order by 字段名 [asc|desc]

order by也可以多字段排序

select \* from my\_student order by class\_id desc , stu\_height asc;

limit 字句

limit 限制子句: 主要用來限制記錄數量獲取。

記錄數限制

純粹的限制獲取數量: 從第一條到指定的數量

語法: limit 數量。

分頁:

利用limit來限制獲取指定區間的數據

limit offset,length;

offset偏移量: 從哪開始。

length長度:獲取多少條數據。

mysql中記錄數量由0開始。

limit 0,2; 表示獲取前兩條記錄。

limit 2,2; 表示獲取由2開始的2條記錄。

PHP提供了三種不同的api連接到MySQL，這三種api是：mysql, mysqli以及PDO擴展。MySql函數是非常受歡迎的,但是官方團隊並不鼓勵再使用它們。在php手冊裡邊，當你看到關於Mysql的函數，會發現都標有紅色的標志，那意味著它們已經被棄用了。棄用ext/mysql函數不僅安全,而且使用其他數據庫查詢方法可以同樣具有MySQL函數的所有功能。更好的方法是使用PDO代替ext/mysql函數。

mysql運算符

算術運算符 + - \* / %

create table ysf1(int\_1 int, int\_2 int, int\_3 int, int\_4 int) charset = big5;

insert into ysf1 values (100,-100,0,default);

*--算術運算*

select int\_1 + int\_2 , int\_1 - int\_2 , int\_1 \* int\_2 , int\_1 / int\_2, int\_2 / int\_3, int\_2 % 6 , int\_4 / 5 from ysf1;

mysql中, 除法用浮點型表示。

如果除法結果為0, 系統會給NULL。

用NULL進行任何算術運算, 結果都為NULL。

比較運算符 (常用在條件中進行限定結果)

> >= < <= = <> <=>

mysql中, 沒有==, 在條件式中, = 就是==。 也可以用<=>。

mysql中, 沒有!=, 在條件式中,!= 就是<>。

select \* from my\_student;

select \* from my\_student where stu\_id <> '0005';

select \* from my\_student where stu\_id = '0005';

select \* from my\_student where stu\_age >=25 ;

特殊應用: 在字段結果中進行比較運算

select '1' <=> 1 , 0.02 <=> 0 , 0.02 <> 0;

mysql中, 數據會先自動轉換成同類型,再比較。

mysql中,沒有true,false。 bool值是用1和0。

在條件判斷時,還有有對應的比較運算符: 計算區間。

between 條件1 and 條件2;

注意: between中 條件1 一定要小於條件2。/\*\*\*

如果是全數字的字符串 放到between中判斷也可以。 系統會自動轉換。

select \* from my\_student where stu\_id between '0001' and '0005';

邏輯運算符

and 邏輯與 or 邏輯或 not邏輯非

in運算符 in 語法

in 在…里面。 用來替代 = , 當結果不是一個值, 而是一個結果時使用。

in (結果1, 結果2, 結果3…); 只要當前條件在結果集中出現過, 那麼就成立。

用於結果數量比較多時。

is 運算符

專門用來判斷值是不是空NULL的。

is null / is not null

\*\*\*\*\*

判斷值是否NULL不能用等號 = null。

like 運算符 (用來進行模糊匹配)

語法: like ‘匹配模式’

匹配模式中, 有兩種占位符,

\_ 匹配單個任何字符

%匹配0到多個任何字符

聯合查詢

合并多個相似的選擇查詢的結果集。等同於將一個表追加到另一個表, 從而實現將兩個表的查詢組合到一起, 使用謂詞union 或 union all。

應用場景

將同一張表中不同的結果(需要對應多條查詢語句來實現), 合并到一起展示數據

最常見: 在數據量大的情況下, 會對表進行分表操作, 需要對每張表進行部分數據統計, 使用聯合查詢來把數據存放到一起顯示。

語法:

select 語句

union [union 選項]

select語句;

[union 選項]: 與select選項基本相同

distinct: 去重複。 (\*\*\* union默認distinct選項)

all:保存所有結果;

注意:

union理論上只要保證字段數一樣, 不需要每次拿到的數據對應字段類型相同。

但不好看。

Order by should be at the very last, after all UNION / UNION ALL statements.

order by一定要放到union後面。

\*\*\*\*

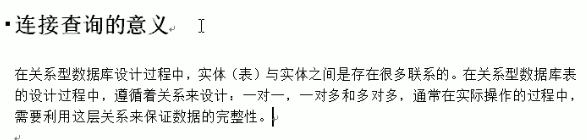
如果兩張表在合并前做order by 要加() 而且要加上limit。

(select \* from my\_student where gender = 1 order by stu\_height desc limit 50)

union

(select \* from my\_student where gender = 2 order by stu\_height asc limit 50);

連接查詢



交叉連接

將兩張表的數據與另外一張表彼此交叉。

原理:

從第一張表依次取出每一條記錄

取出每一條記錄後,與另外一張表的全部記錄依次匹配

沒有任何匹配條件,所有結果都會保留

記錄數= 第一張表記錄數\* 第二張表記錄數;

字段數 = 第一張表字段數 + 第二張表字段數;

(笛卡爾積) 不是好東西。 實用價值極低\*\*\*\*\*。

語法:

表1 cross join 表2;

應用:

corss join結果是(笛卡爾積) 沒有實用價值

結果可以用 corss join也可以用 from 表1, 表2 實現。

內連接

inner join 從一張表取出數據去匹配另一張表,

匹配成功則保留。

語法

表1 [inner] join 表2 on 匹配條件;

如果inner join 沒有條件, 就會得到笛卡爾積, cross join的結果。

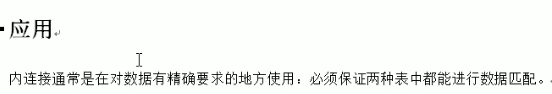
因為不同表之間通常出現同字段名, 所以我們可以用 表名.字段名, 確保唯一性。

select \* from my\_student inner join my\_class on my\_student.class\_id = my\_class.id;

select \* from my\_student as a inner join my\_class as b where a.class\_id = b.id;

inner join 用where 與on 都可以。

inner join 的應用



外連接

outer join



語法:

左連接: 主表 left join 從表 on 連接條件;

右連接: 從表 right join 主表 on 連接條件;

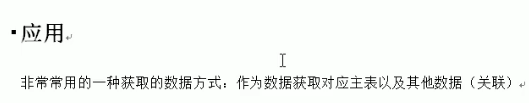
左連接對應的主表數據在左邊。

右連接對應的主表數據在右邊。

\*\*\*

主表會保留所有數據顯示(即使匹配不成功)。

outer join 的應用



自然連接

\*\*\*\*\*

通過MySql自己的判斷完成連接過程，不需要指定連接條件。MySql會使用表內的，相同的字段，作為連接條件。

自然連接分為內外之分。

*--先把my\_class的 id 改為class\_id;*

alter table my\_class change id class\_id int;

內連接 natural join

select \* from my\_class natural join my\_student;

外連接　natural join

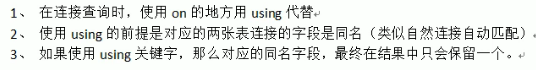
select \* from my\_class natural left join my\_student;

select \* from my\_class natural right join my\_student;

using 關鍵字 (合并同字段名的結果顯示)

在連接查詢中, 代替 on 關鍵字, 進行條件匹配。

原理:



語法:

表1 [inner,left,right] join 表2 using (同名字段名列表); --連接字段

select \* from my\_student left join my\_class using(class\_id);

子查詢

sub query

是一種常用計算機語言 select-sql 語言中嵌套查詢下層的程序模塊, 當一個查詢是另一個查詢的條件時, 就是子查詢。



子查詢分類

標量子查詢 返回結果是一個數據(一行一列)

列子查詢 返回結果是一列(一列多行)

行子查詢 返回結果是一行(一行多列)

表子查詢 返回結果是多行多列(多行多列)

Exists子查詢 返回結果0或1(類似bool)

按位置分

where子查詢 : 子查詢出現的位置在where條件中

from子查詢 : 子查詢出現的位置在from數據源中, (做數據源)

標量子查詢

返回結果是一個數據(一行一列)

select\* from數據源 where條件判斷 = / <> (select 字段名

from 數據源 where 條件判斷); -- 子查詢得到的結果只有一個值

select \* from my\_class where class\_id = (select class\_id from my\_student where stu\_name = '小美');

列子查詢

返回結果是一列(一列多行)

語法:

主查詢 where 條件 in (列子查詢)

因為有多個列子查詢結果 所以用in。

行子查詢

返回結果是一行(一行多列)

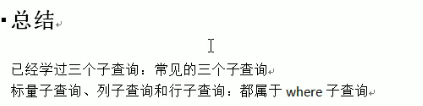
行元素: 字段元素是指一個字段對應的值, 行元素對應的就是多個字段: 多個字段合起來作為一個元素參與運算, 就是行元素。

行元素 類似 字段名列表

語法:

主查詢 where 條件 [ (構造一個行元素) ] = (行子查詢);

總結



表子查詢

返回結果是多行多列(多行多列)

行子查詢 是用於where條件判斷: where子查詢

表子查詢 是用於from數據源: from子查詢

語法:

select 字段名列表 from (表子查詢)as別名 [where] [group by] [having] [order by] [limit]。

\*\*\*\*\*

如果ORDER BY列有多行具有相同的值，服務器可以自由地以任何順序返回這些行，並且根據總體執行計劃可能以不同的方式返回。換句話說，這些行的排序順序對於無序列是不確定的。

影響執行計劃的一個因素是LIMIT，因此對於一個ORDER BY查詢而言，帶與不帶LIMIT返回的行的順序可能是不一樣的。

order by排序如果出錯 加limit 常常可解決問題。

select \* from (select \* from my\_student order by stu\_height desc limit 100) as temp group by class\_id;

如果你需要確保無論帶不帶LIMIT都要以相同的順序返回，那麼你可以在ORDER BY中包含附加列，以使順序具有確定性。例如：

exists 子查詢

返回結果是0 或1。 0 = false 1=true;

語法:

where exists (子查詢)

如果存在 返回1

不存在 返回0

where 1 永遠為真。

*--求出 有學生在的所有班級*

select \* from my\_class as c where exists (select stu\_id from my\_student as s where s.class\_id = c.class\_id);

子查詢中特定關鍵字的使用

in

主查詢 where 條件 in (列子查詢)

any

任意一個

= any (列子查詢), 條件在(查詢結果列)中任意一個匹配即可。 等價於in

<> any (列子查詢) 條件不等於(查詢結果列)中任意一個匹配。

1 = any (1,2,3,4) ===== true;

1 <> any (1,2,3,4) =====true; 不等於任意一個。

some

與any完全一樣。

all

= all (列子查詢); 等於里面所有

<> all(列子查詢); 不等於其中所有

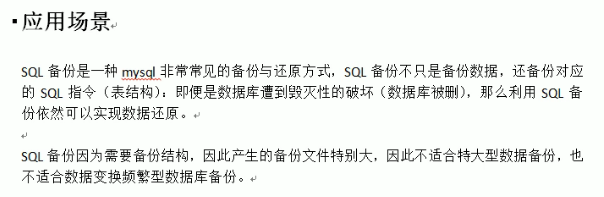
如果對應的字段名的值有NULL, 那麼不參與匹配。

整庫數據備份與還原

整庫數據備份也叫SQL數據備份, 備份的結果都是SQL指令。

在mysql中 bin文件內mysqldump.exe 就是用來備份SQL的。

應用場景



語法:

mysqldump / mysqldump.exe –h… -P… -u… -p… 數據庫名字 [表1 [表2…]] > 備份文件地址/備份名.sql 不需要;號結尾 在cmd登錄mysql前使用。

mysqldump.exe -hlocalhost -P3306 -uroot -proot mydatabase2 > "C:/ProgramData/MySQL/MySQL Server 8.0/Data/temp/mydatabase2.sql"

備份有3種形式:

1. 整庫備份 (只需要數據庫名字)

mysqldump.exe -hlocalhost -P3306 -uroot -proot mydatabase2 > "C:/ProgramData/MySQL/MySQL Server 8.0/Data/temp/mydatabase2.sql"

1. 單表備份 (數據庫名字後跟表名)
2. 多表備份 (數據庫名字後跟多張表名)

mysqldump.exe -uroot -proot mydatabase2 my\_student my\_int > "C:/ProgramData/MySQL/MySQL Server 8.0/Data/temp/student\_int.sql"

數據還原

\*\*\*在還原之前, 我們要先開個空數據庫去放想要還原的內容。

2種方式還原:

1. 利用mysql.exe客戶端, 在沒有登錄之前, 可以直接用客戶端進行數據還原

mysql.exe –h…-P…-u…-p… 數據庫 < 文件位置

1. 在SQL指令, 提供了一種導入SQL指令的方式

source SQL 文件位置 (必須先進入到空的數據庫/想要還原的數據庫)

1. 人為操作, 打開備份文件, 複制所有SQL指令. 到mysql.exe客戶端中去直接輸入指令(不推薦)

登錄之前的操作不要用 ;號。\*\*\*

mysql.exe -uroot -proot mydatabase2 < "C:/ProgramData/MySQL/MySQL Server 8.0/Data/temp/mydatabase2.sql"

source C:/ProgramData/MySQL/MySQL Server 8.0/Data/temp/mydatabase2.sql;

drop table my\_int , my\_student;

source C:/ProgramData/MySQL/MySQL Server 8.0/Data/temp/student\_int.sql;

用戶權限管理

在不同項目中給不同的角色(開發人員)不同的操作權限。

通常,一個用戶密碼不會長期不變。

所以需要更變數據庫用戶密碼來保證用戶安全(mysql客戶端 用戶)

用戶管理

select \* from mysql.user\G



在mysql中,對用戶管理中, 是由對應的host與user共同組成主鍵來區分用戶。

user: 代表用戶的用戶名

host: 代表本質是允許訪問的客戶端(IP或者主機地址)。 如果host是 \* 號,代表所有的用戶(客戶端)都可以訪問。

創建用戶

2種方式:

1. 直接使用root用戶在mysql.user表中插入記錄 (不推薦)
2. 利用創建用戶SQL指令

create user 用戶名 identified by ‘明文密碼’;

用戶名: 用戶名@主機地址

主機地址: ‘ ‘ (空) or ‘%’ (%是匹配0到多任何字符串)

create user 'user1'@'%' indentified by '123456';

‘%’是任意的主機地址

簡化的創建用戶

create user ‘user2’;

這個用戶沒有密碼, 不限定客戶端ip (不安全)。

刪除用戶

mysql中user是帶著host的(具有唯一性)

語法:

drop user 用戶名@host;

修改用戶密碼

使用系統函數 password(),  mysql8.0不需要用password函數了

1.set password for 用戶名 = '新的明文密碼';

set password for 'user1'@'%' = '654321';

\*\*\*\*\*

set password for 'root'@'localhost' = '123456';

我們改了root的密碼

權限管理

MYSQL權限管理分為3類

1. 數據權限: crud (select/update/delete/insert)
2. 結構權限: 結構操作(create / drop)
3. 管理權限: 權限管理(create user / grant / revoke) 通常只給管理員

授予權限 grant … to

grant 權限列表 on 數據庫.表名 to 用戶

數據庫.\*是該數據庫內所有表 \*.\* 是所有數據庫數有表

權限列表: 使用 , 號分隔, 但是可以使用 all privileges 代表全部權限

grant select on mydatabase2.\* to 'user1'@'%';

取消權限 / 權限回收 : revoke …. from

revoke 權限列表 on 數據庫.表名 from 用戶

刷新權限 flush

flush: 刷新, 將當前對用戶的權限操作, 進行一次刷新, 將操作的具體內容同步到對應的表中。

flush privileges;

找回丟失密碼

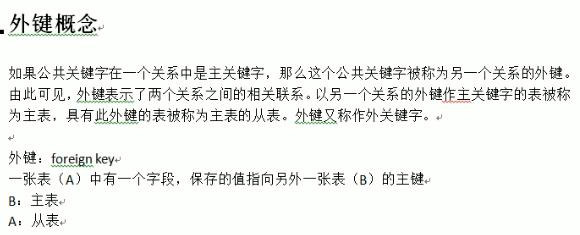
如果忘記了root 用戶密碼, 就需要去找回或重置root用戶密碼

1. 停止mysql服務 net stop mysql80;
2. 重新啟動服務 mysqld.exe --skip-grant-talbes //啟動服務器但是跳過權限
3. 當前啟動的服務器沒有權限概念, 十分危險, 任何客戶端, 不需要任何用戶信息都可以直接登錄, 而且是root權限,
4. 新開客戶端, 使用mysql.exe登錄即可,不需要-u -p。
5. 修改root用戶密碼, 指定 用戶名@host

update mysql.user set password = ‘newpassword’ where user=’root’ and host = ‘localhost’;

1. 馬上關服務器(當前無密碼的mysql.exe cmd窗口)
2. 重啟服務 net start mysql80;

外鍵foregin key



外鍵的操作

增加外鍵 foreign key

兩種方式增加外鍵:

1. 在創建表時增加外鍵(類似主鍵)

在字段名列表後增加一條語句

[constraint `外鍵名`] foreign key(外鍵字段) references 主表(主鍵);

create table my\_foreign (

id int primary key auto\_increment,

name varchar(10) not null,

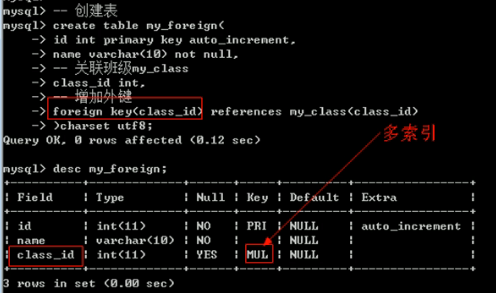
*--關聯班級 my\_class*

class\_id int ,

*--增加外鍵*

foreign key(class\_id) references my\_class(class\_id)

) charset = big5;



key : MUL 多索引, 外鍵本身是一個索引,外鍵要求外鍵字段本身也是一種普通索引。

show create table my\_foreign;



1. 在創建表後增加外鍵

alter table 從表 add [constraint `外鍵名`] foreign key(外鍵字段) references 主表(主鍵);

`外鍵名` = `student\_class\_ibfk\_1`

alter table my\_student add constraint `student\_class\_ibfk\_1` foreign key(class\_id) references my\_class(class\_id);

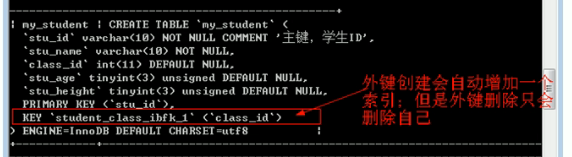
alter table my\_student add foreign key(class\_id) references my\_class(class\_id);

修改 & 刪除外鍵

外鍵不能修改, 只能先刪除,再增加。

alter table 從表 drop foreign key `外鍵名`;

外鍵不能刪除自動產生的普通索引, 只會刪除外鍵自己。



如果想刪除對應索引:

alter table 表名 drop index 索引名字;

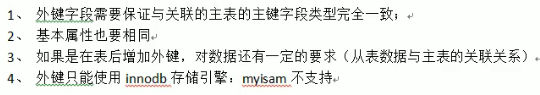
刪除外鍵:

alter table my\_student drop foreign key `student\_class\_ibfk\_1`;

刪除對應索引:

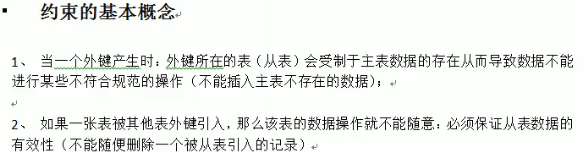
alter table my\_student drop index `student\_class\_ibfk\_1`;

外鍵基本要求



外鍵約束

通過建立外鍵關系後, 對主表和從表都會有一定的數據約束效率。



*向從表my\_foreign 插入數據*

可以在創建外鍵時, 對外鍵約束進行選擇性的操作。

約束模式

語法:

add foreign key (外鍵字段) references 主表(主鍵) on 條件 約束模式;

district: 嚴格模式 , default; (不允許操作)

cascade: 級聯模式 一起操作(主表變化,從表數據跟著變)

set null: 置空模式(主表變化(刪除), 從表對應記錄設置為空: 前提是從表對應的外鍵字段允許為空)

\*\*\*

外鍵約束主要約束的對象是主表操作, 從表就是不能插入主表不存在的數據。

進行約束時, 需要指定操作 update和delete

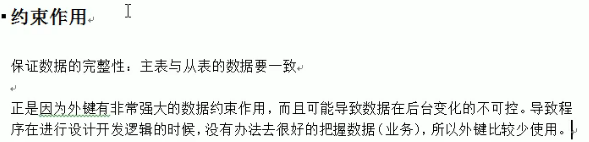
常用的約束模式

on update cascade; 更新級聯

on delete set null; 刪除置空

alter table my\_student add foreign key(class\_id) references my\_class(class\_id) on update cascade on delete set null;

外鍵約束的作用



視圖基本操作

創建視圖

語法:

create view 視圖名字 as select 指令;

可以是單表數據,連接查詢,聯合查詢,子查詢

create view student\_class\_ibfk\_v as

select s.\* , c.name from my\_student as s left join my\_class as c on s.class\_id = c.class\_id;

查看視圖結構

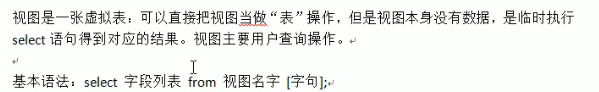
show tables ;

show create table [view]

desc 視圖名字;

使用視圖

select \* from視圖名字;



修改視圖

本質是修改視圖對應的查詢語句

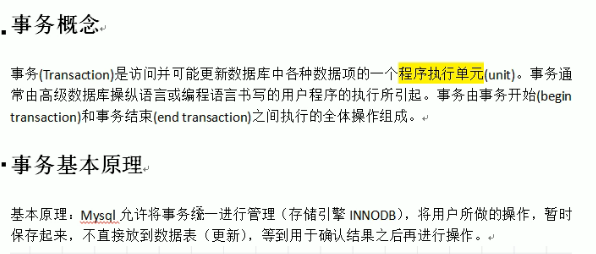
alter view 視圖名 as 新 select 指令;

刪除視圖

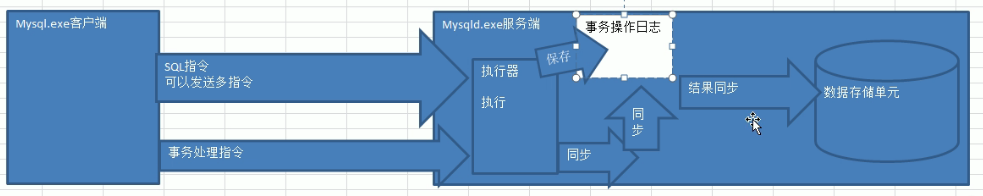
drop view視圖名;

事務安全

事務 transaction



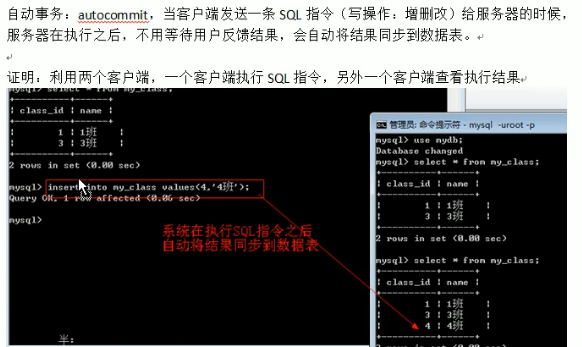
可以等到想結果同步時, 再 invoke 使用事務處理指令。



事務在mysql中通常是自動提交的, 但是也可以使用手動事務。

自動事務

autocommit,



show variables like 'autocommit%';



關閉自動事務

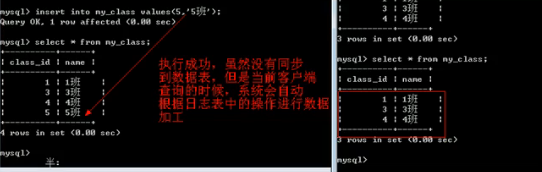
set autocommit = Off;

自動事務一旦關閉,那麼需要用戶提供是否同步的命令

commit 提交(同步到數據表)

rollback 回滾(清空之前的操作, 不要了)

事務沒有提交的對比查看



通常不會關閉自動事務。

最多是需要手動事務時用手動事務。

手動事務

不管是開始還是過程還是結束都需要用戶(程序員), 手動的發送事務操作指令來實現。

手動事務的命令:

1.start transcation; 開啟事務, 從這條語句開始, 後面的所有語句都不會直接寫入到數據表(保存在事務日志中)

2.事務處理: 多個寫指令構成

3.事務提交: commit / rollback 到這個時候所有的事務才算結束。

回滾點

savepoint, 當有一系列事務操作時, 而其中的步驟如果成功了, 沒有必要重新來過, 可以在某個點(成功), 設置一個記號(savepoint), 然後如果後面有失敗, 可以回到這個savepoint。

語法:

增加savepoint: savepoint 回滾點名字;

回到savepoint: rollback to 回滾點名字;

如果有多個回滾點, 回到前面的回滾點時, 後面的回滾點就會失效。

如果在start transction;的過程中, 不小心用了另一次start transaction; 會自動commit;所有東西。

*--開啟事務*

start transaction;

*--執行事務*

select \* from my\_student;

select \* from my\_class;

insert into my\_class values(6,'F班');

update my\_student set class\_id = 6 where stu\_id = '0007';

*--提交事務*

commit;

*--滾回事務 rollback*

start transaction;

insert into my\_class values(7,'G班');

select \* from my\_class;

rollback;

*--回滾點 savepoint*

select \* from my\_class;

start transaction;

insert into my\_class values(100,'ZZ班');

savepoint sp1;

insert into my\_class values(200,'ZZZ班');

rollback to sp1;

commit;

事務安全- 特性

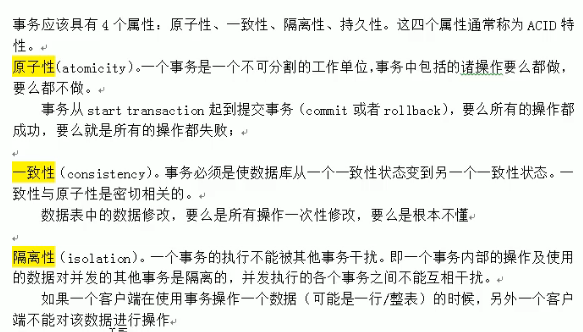
ACID 四個特性

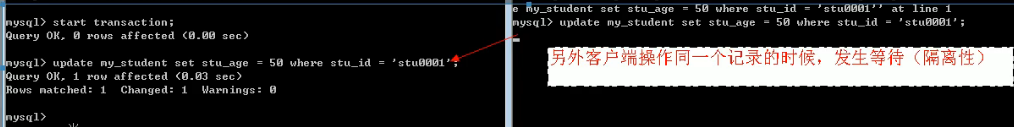
原子性 atomicity

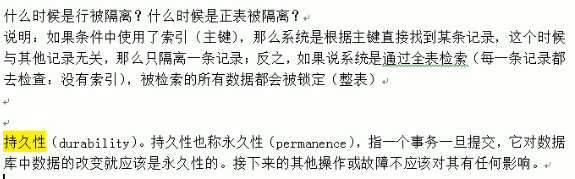
一致性 consistency

隔離性 isolation

持久性 durability



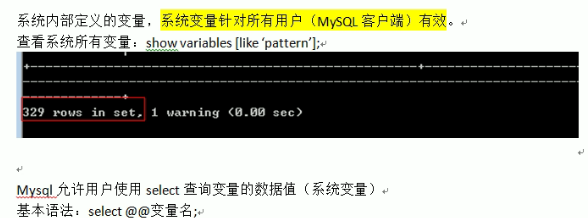




變量



系統變量

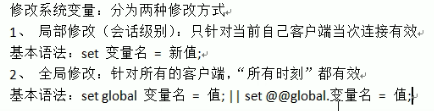


*--查看系統變量*

select @@autocommit;

@@是全局的意思

修改系統變量



1.局部修改:(會話級別 = 只在當前客戶端有效)

set 變量名 = 新值;

2.全局修改:

set global 變量名 = 值;

set @@global 變量名 = 值;

--會話修改系統變量

set autocommit = 0;

局部修改:(會話級別 = 只在當前客戶端有效)

set autocommit = 0;

全局修改:

set global autocommit = 0;

set @@global.autocommit = 0;

\*\*\*\*\*

全局修改: 所有連接的客戶端沒有變改?

因為全局修改只針對新客戶端生效。(正在連接的無效)

show variables like 'character\_set%';

set names big5;

SET global character\_set\_client = big5;

SET global character\_set\_results = big5;

SET global character\_set\_connection = big5;

set global character\_set\_client = utf8;

set global character\_set\_results = utf8;

set global character\_set\_connection = utf8;

會話變量(用戶變量): 只對當前用戶使用的客戶端生效



由於mysql沒有 == 號, 所以賦值和比較運算有時會出錯, 所以mysql推出了一個新的變量賦值符號 := 。

定義用戶變量

set @變量名 = 值;

set @變量名 := 值; 我們全部的變量賦值都應該用 :=

mysql是專門存儲數據的, 充許將數據從表中取出存儲到變量中, 查詢得到的數據必須只能是一行數據(一個變量對應一個字段值) mysql沒有數組。

1. 賦值且查看賦值過程: select @變量1 := 字段1 , @變量2 := 字段2 from 數據表 where 條件;

select @name := stu\_name , @age := stu\_age from my\_student limit 1;

1. 只賦值, 不看過程:

select 字段1 , 字段2 … from 數據源 where 條件 into @變量1 , @變量2…

查看自定義變量

select @變量名;

局部變量

作用範圍在begin到end語句塊之間, 在該語句塊里設置的變量, declare語句專門用於定義局部變量。

1. 局部變量使用declare關鍵字聲明
2. delacre語句出現的位置一定在begin與end之間。(begin end 是在大型語句塊中使用, 函數/存儲過程/觸發器)
3. 聲明語法: declare 變量名 數據類型 [屬性];

流程結構

if語句

mysql中有2種方式使用if:

1. 用在select查詢中, 當做一種條件來進行判斷

if(條件 , 為真結果 , 為假結果)

select \* , if(stu\_age >=20 ,'符合' , '不符合') as test from my\_student;

select \* , if(條件 , 為真結果 , 為假結果) as if結果字段名 from 數據源;

1. 用在複雜語句塊中(函數/存儲過程/觸發器)

if 條件表達式

then

滿足條件時執行的語句

end if;

複合語法

代碼的判斷存在兩面性, 兩面都有對應的代碼執行。

語法:

if 條件表達式

then滿足條件時執行的語句

else 不滿足條件時執行的語句

end if;

while循環

mysql的循環體都是在大型代碼塊中使用

語法:

while 條件 do

循環執行代碼

end while;

結構標識符

為某些特定的結構進行命名, 為的是在某些地方使用這名字。

語法:

標識名字: while條件 do

循環執行代碼

end while[標識名字];

while結構標識符 主要是為了循環體中使用循環控制。

mysql中,沒有break 沒有continue; 但有自己的關鍵字替代。

iterate: 疊代, 就是 continue;(以下代碼不執行, 重新開始循環)

leave : 就是break; (整個循環中止)

標識名字: while條件 do

if 條件判斷 then

循環控制;

iterate / leave 標識名字;

end if;

循環體

end while[標識名字];

函數

mysql有2種函數: 系統函數(內置函數) 與 自定義函數。

\*\*\* 不管是內置函數 還是 用戶自定義函數,

都是使用 select 函數名(參數列表);

常見的系統函數(內置函數)

字符串函數

char\_length()

length()

concat()

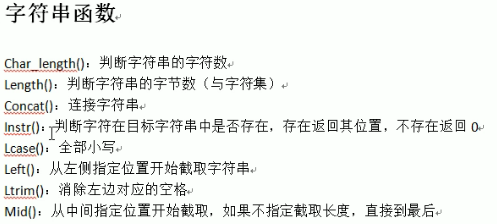
instr()

lcase()

left()

ltrim()

mid()



時間函數

now()

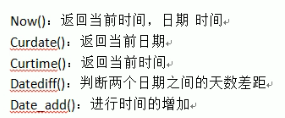
curdate()

curtime()

datediff(): 參數日期需要字符串格式

date\_add(日期,interval時間數字type): 進行時間的增加

type: day/hour/minute/second



unix\_timestamp(); 返回時間戳(秒)

from\_unixtime(second); unix秒轉換做date+time

select from\_unixtime(unix\_timestamp()) , curdate() , curtime();

數學函數

abs()

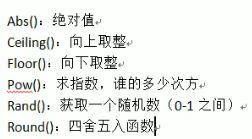
ceiling()

floor()

pow()

rand() 0到1之間。

round()



其他函數

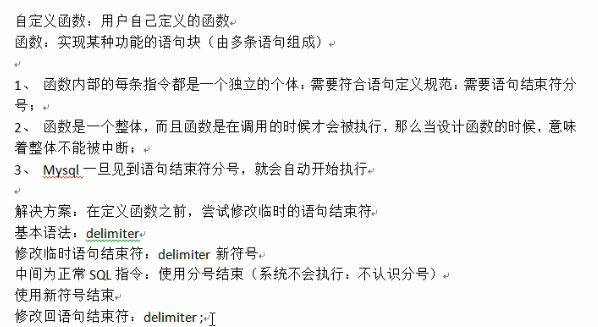
md5(); 對數據進行md5加密

version();當前mysql version

database(); 顯示當前數據庫

uuid(); 生成一個唯一標識符(自增長): 自增長是單表唯一, UUID是整庫(數據唯一同時空間唯一)

自定義函數



delimiter 新符號[$$] 修改臨時語句結束符

修改回結束符 delimiter;

創建函數

自定義函數內容: function 關鍵字 , 函數名 , 參數 (形參 / 實參 (可選)), 確認函數返回值類型, 函數體 , 返回值。

語法:

修改語句結束符

create function 函數名 (參數) returns 類型

begin

//函數體

return 返回值數據;

end

語句結束符

修改語句結束符(改回來)

set global log\_bin\_trust\_function\_creators = true;

*-- 解鎖delimiter 可使用*

delimiter $$

create function my\_func1() returns int

begin

    return 10;

end

$$

delimiter ;

最簡函數

如果只有一條, 就不需要begin和end

create function my\_func2() returns int

return 20;

形參: 在mysql中 需要為函數的形參指定數據類型(形參可以有多個)

基本語法: 變量名 字段類型

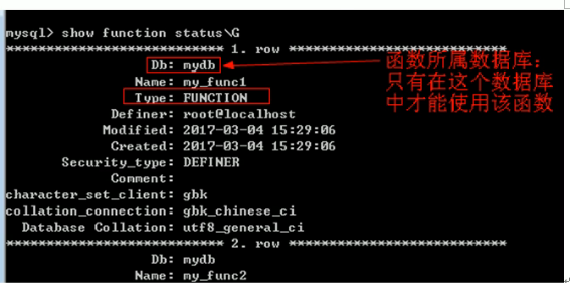
create function my\_func3(int\_1 int , int\_2 int) returns int

return int\_1 + int\_2;

查看函數

show function status [like ‘pattern’];

show function status like ‘my\_func%’\G;



\G 的作用是將查到的結構旋轉90度變成縱向：

--查看函數創建語句

show create function my\_func1;

調用函數

select 函數(參數);

select my\_func3(10,20);

刪除函數

drop function 函數名;

1.自定義函數是會話級別(用戶級別), 只在當前客戶端對應的數據庫有效。

2.只能在對應數據庫調用。

3.自定義函數通常是為了將多行代碼集合到一起解決一個重複性的問題。

4.函數因為必須規範返回值: 那麼在函數內部不能使用select指令: select一旦執行就會得到一個結果(result set)。 (唯一可用: select 字段 into @變量;)

函數案例:

需求: 由1開始, 直到用戶傳入的對應的值為止 ,自動求和, 凡是5的倍數都不要。

局部變量

declare 變量名 類型 [=默認值];

delimiter $$

create function exec1(int\_f int) returns int

begin

    declare result int default 0;

    declare i int default 1;

    mywhile:while i < int\_f do

        if i = 5 then

            set i = i + 1;

            iterate mywhile;

        end if;

        set result = result + i;

        set i = i + 1;

    end while mywhile;

    return result;

end

$$

delimiter ;

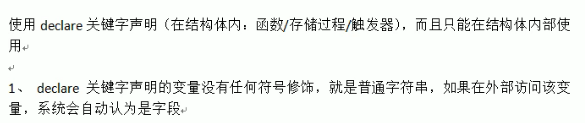
mysql 修改變量要用set

mywhile 標識符

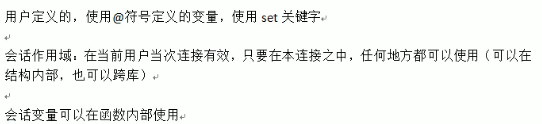
iterate 就是 continue;

變量作用域

局部作用域

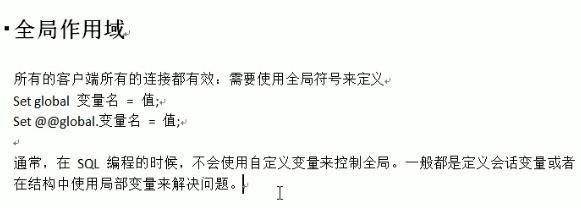


會話作用域

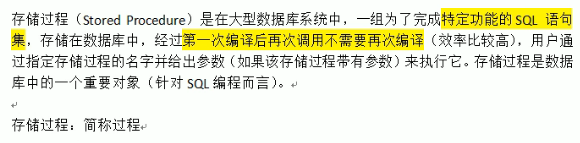


會話變量可以跨庫

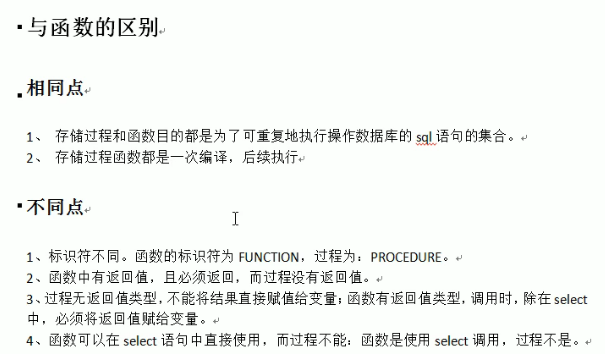
全局作用域



存儲過程 stored procedure



過程與函數的別區



存儲過程的操作

創建過程

create procedure 過程名字([參數列表])

begin

過程體

end

結束符

如果過程體只有一條, 可以不寫begin end。

\*\*\* 過程基本上可以完成所有函數的功能。\*\*\*

delimiter $$

create procedure my\_pro2()

begin

    declare i int default 1;

    set @sum = 0;

    while i <=100 do

    set @sum = @sum + i;

    set i = i + 1;

    end while;

    select @sum;

end

$$

delimiter ;

查看過程

show procedure status [like ‘pattern’];

show procedure status like 'my\_pro%';

show procedure status like 'my\_pro%'\G

show create procedure my\_pro2;

調用過程

過程沒有返回值, 不能用select 調用

call 過程名(實參);

刪除過程

drop procedure 過程名;

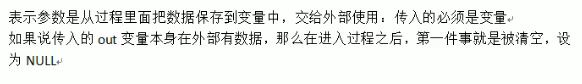
存儲過程的形參類型



in



out



inout



形參使用語法

過程類型 變量名 數據類型

會話變量可以自由在過程/函數 內外部調用invoke。

*--創建有參過程*

delimiter $$

create procedure my\_pro3(in int\_1 int , out int\_2 int , inout int\_3 int)

begin

    select int\_1 , int\_2 , int\_3;

    set int\_1 = 10;

    set int\_2 = 100;

    set int\_3 = 1000;

    select int\_1 , int\_2 , int\_3;

*-- 會話變量*

    select @n1 , @n2 , @n3;

    set @n1 = 'a';

    set @n2 = 'b';

    set @n3 = 'c';

    select @n1 , @n2 , @n3;

end

$$

delimiter ;

call my\_pro3(@n1,@n2,@n3);

select @n1 , @n2 , @n3;

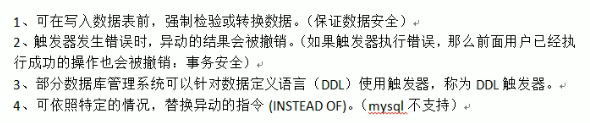
*--      a  , 100 , 1000*

*--  @n2 和 @n3 分別是out 和 inout 過程類型, 所以會複蓋值*

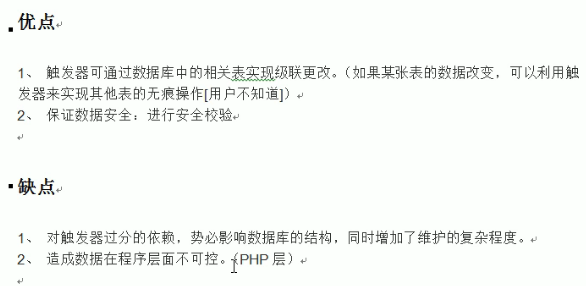
觸發器



作用



觸發器的優缺點



觸發器基本語法

創建觸發器

create trigger 觸發器名字 觸發時機 觸發輸件 on 表 for each row

begin

end

觸發對象: on 表 for each row, 觸發器定實質是表中的所有行, 因此當每一行發生指定的改變時, 就會觸發觸發器。

觸發時機: 每張表中對應的行都會有不同狀態, 當SQL指令發生時,

都會令行中發生數據改變, 每一行總會有兩種狀態, 數據操作前 和 操作後。

before: 表中數據操作前的狀態

after: 表中數據操作後的狀態

觸發事件: mysql中觸發器針對目標是數據發生改變 , 對應的操作只有寫操作(增刪改)

insert 插入操作

update 更新操作

delete 刪除操作

注意事項:

一張表中, 每一個觸發時機 綁定事件對應的觸發器類型 只能有一個。

一張表中只能有一個對應after insert的觸發器。

因此, 一張表最多只有2x3 = 6 種/個觸發器。

需求: 有兩張表 , 一張是商品表 , 一張是訂單表(保留商品ID) , 每次訂單生成 , 商品中對應的庫存就應該發生變化。

查看觸發器

show triggers;

show triggers like '%\_t';

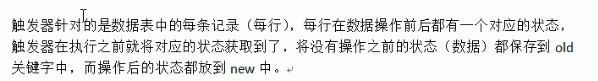
show create trigger after\_insert\_order\_t;

刪除觸發器

drop trigger觸發器名字;

觸發器應用

記錄關鍵字 new old



觸發器中可以通過 old 和 new獲取綁定表中對應的記錄數據。

語法:

關鍵字.字段名

insert: 插入前為空, 沒有old

delete: 清空後為空, 沒有new

update: old new 都有。

*--重寫自動扣除商品庫存的觸發器*

delimiter $$

create trigger after\_insert\_order\_t after insert on my\_orders for each row

begin

    update my\_goods set inv = inv - new.goods\_num where id = new.goods\_id;

end

$$

delimiter ;

*--觸發觸發器*

insert into my\_orders values (null,1,5);

insert into my\_orders values (null,2,10);

insert into my\_orders values (null,3,3);

select \* from my\_orders;

select \* from my\_goods;

*-- 判斷庫存*

delimiter $$

create trigger before\_insert\_order\_t before insert on my\_orders for each row

begin

   select inv from my\_goods where id = new.goods\_id into @inv;

   if @inv < new.goods\_num then

*--主動出錯, 觸發器出錯就會取消操作*

        insert into xxx values('xxx');

   end if;

end

$$

delimiter ;

*--觸發觸發器  判斷庫存*

insert into my\_orders values (null,3,1000);