MySQL簡易教程

1. 此教程無包含安裝過程，請自行安裝
2. 教程內容使用MySQL Workbench，但與使用MySQL Shell編輯、下指令方式無異

建功高中物理科
cow3690m@tmail.hc.edu.tw

呂昀修

目錄

Chap. 1
表格介紹

建功高中物理科
cow3690m@tmail.hc.edu.tw

呂昀修

Chap. 1 範例表格

Employee

emp_id	name	birthday	gender	salary
206	小黑	1999/10/8	F	50000
207	小白	1985/9/16	M	30000
208	小綠	2000/12/19	M	35000
209	小藍	1997/1/22	F	47000
210	小黃	1990/11/10	M	68000

header name
具有唯一性

row

primary key
具有唯一性

column

建功高中物理科
cow3690m@tmail.hc.edu.tw
呂昀修

Chap. 1 表格間的連結（外鍵foreign key）

foreign key

Employee

emp_id	name	birthday	gender	salary	branch_id
206	小黑	1999/10/8	F	50000	1
207	小白	1985/9/16	M	30000	2
208	小綠	2000/12/19	M	35000	3
209	小藍	1997/1/22	F	47000	1
210	小黃	1990/11/10	M	68000	3

必須對應到另一個表格的primary key

Branch

branch_id	name
1	研發
2	行政
3	資訊

Chap. 1 表格間的連結（外鍵foreign key）

foreign key

Employee

emp_id	name	birthday	gender	salary	branch_id
206	小黑	1999/10/8	F	50000	1
207	小白	1985/9/16	M	30000	2
208	小綠	2000/12/19	M	35000	3
209	小藍	1997/1/22	F	47000	1
210	小黃	1990/11/10	M	68000	3

Branch

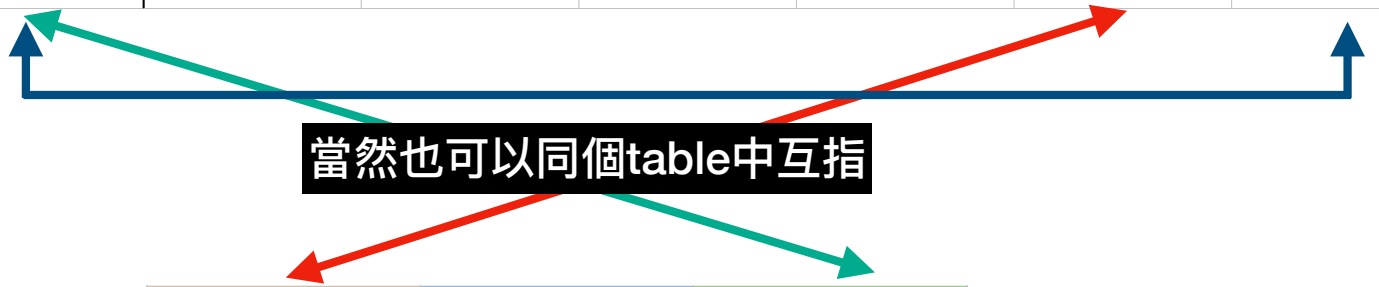
branch_id	name	manager_id
1	研發	206
2	行政	207
3	資訊	208

Chap. 1 表格間的連結（外鍵foreign key）

foreign key

Employee

emp_id	name	birthday	gender	salary	branch_id	buddy_id
206	小黑	1999/10/8	F	50000	1	NULL
207	小白	1985/9/16	M	30000	2	NULL
208	小綠	2000/12/19	M	35000	3	NULL
209	小藍	1997/1/22	F	47000	1	206
210	小黃	1990/11/10	M	68000	3	208



Branch

branch_id	name	manager_id
1	研發	206
2	行政	207
3	資訊	208

Chap. 1 多主鍵（無法以單個primary key表示唯一資料時）

Employee

emp_id	name	birthday	gender	salary	branch_id	buddy_id
206	Alice	1999/10/8	F	50000	1	NULL
207	Bob	1985/9/16	M	30000	2	NULL
208	Herry	2000/12/19	M	35000	3	NULL
209	Eve	1997/1/22	F	47000	1	206
210	John	1990/11/10	M	68000	3	208

Branch

branch_id	name	manager_id
1	研發	206
2	行政	207
3	資訊	208

Client

client_id	client_name	phone
400	Apple	3412
401	Banana	55688
402	Candy	4022
403	Dino	34157
404	ETE	8864

Works_with

emp_id	client_id	total_sales
206	**400**	70000
207	**401**	24000
208	**400**	10000
208	**403**	24000
210	**404**	88000

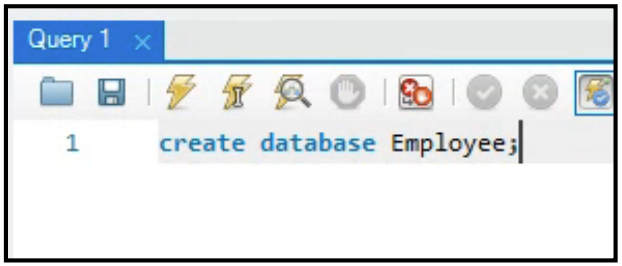
Chap. 2
創建資料庫（database）

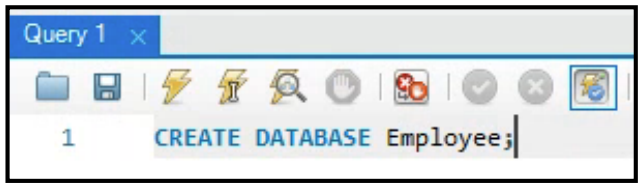
建功高中物理科
cow3690m@tmail.hc.edu.tw

呂昀修

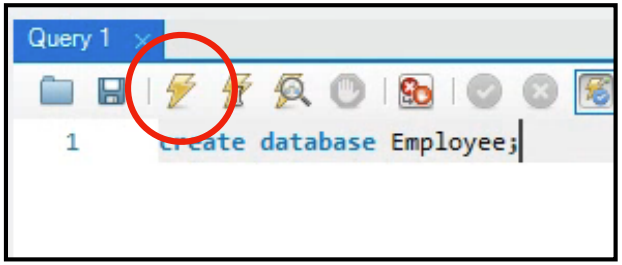
Chap. 2 建立資料庫（create database `name`;）

大小寫都沒差
以前的SQL規定指令都要大寫只是因為比較好分辨

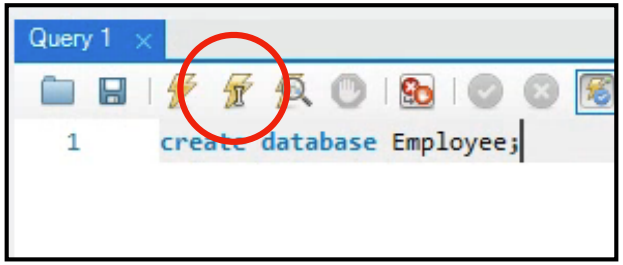




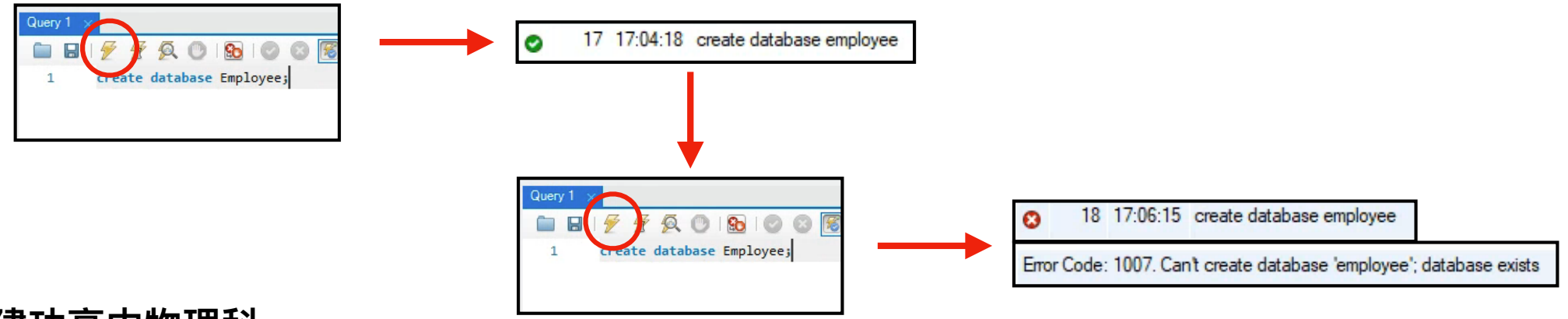
執行全文件



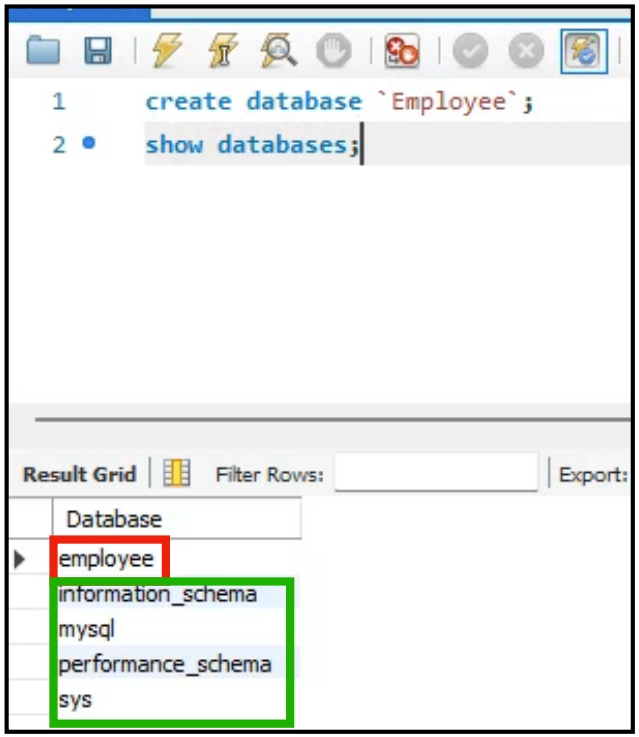
執行單行指令



當執行過某指令後，你的server便會紀錄當前狀態，並不像其他程式一樣會重來一次。
也就是說，已經建立的table，是不可以再建立一次的！



Chap. 2 顯示資料庫（show databases;）



新創建的

預設的

會發現不論你指令中打大小寫，database都是小寫

Chap. 2 刪除資料庫（drop database `name`;）

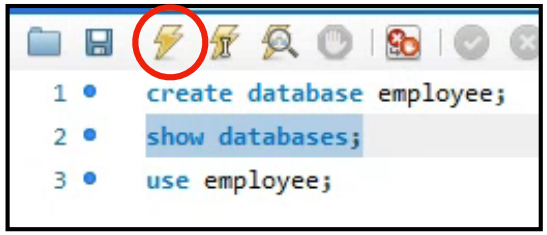
```
3 •  drop database employee;
4 •  show databases;
```

Result Grid | Filter Rows:

Database
information_schema
mysql
performance_schema
sys

同理，不可以刪除不存在的table

Appendix



當我們只想要執行「某幾行」指令時，只需要反白該幾行，並且按執行即可

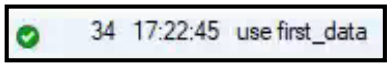
Chap. 3
創建表格（table）

建功高中物理科
cow3690m@tmail.hc.edu.tw

呂昀修

Chap. 3 使用資料庫（use `name`;）

```
create database `first_data`;
show databases;
use `first_data`;
```



這樣才算是有使用成功！

Chap. 3 創建表格（create table `name` (~);）

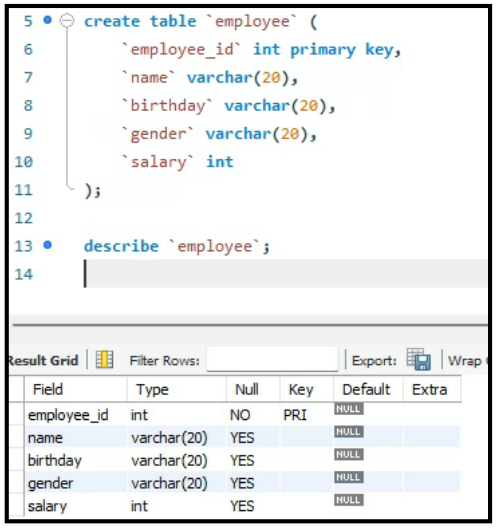
```
create table `employee` (
    `employee_id` int primary key,
    `name` varchar(20),
    `birthday` varchar(20),
    `gender` varchar(20),
    `salary` int
);

describe `employee`;
```

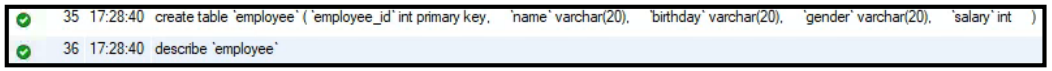
以前面employee表格為例
先完成左側的內容

describe `employee`指的是
顯示出employee這個表格
的內容

與左側等價

```
create table `employee` (
    `employee_id` int,
    `name` varchar(20),
    `birthday` varchar(20),
    `gender` varchar(20),
    `salary` int,
    primary key(`employee_id`)
);
describe `employee`;
```

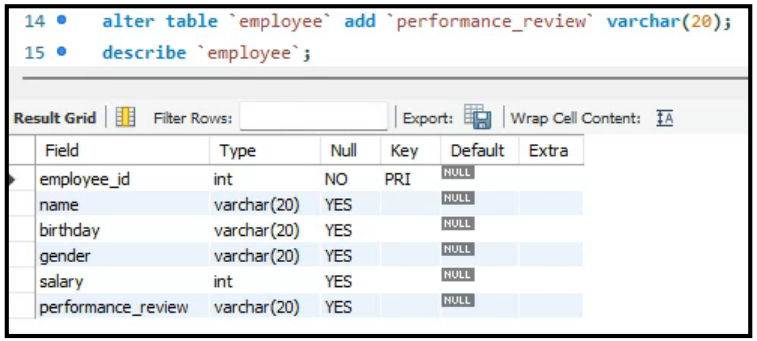




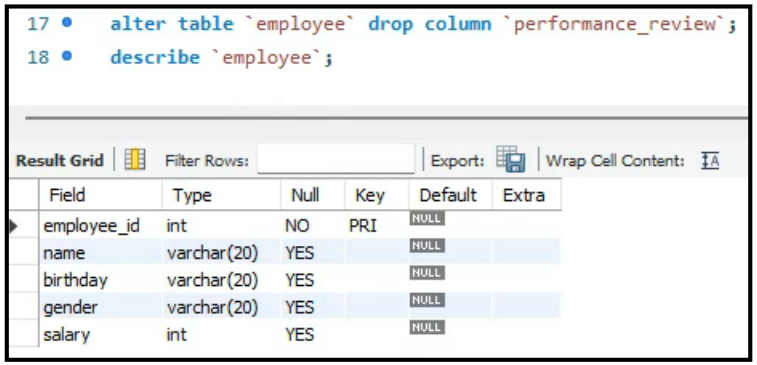
執行結果

Chap. 3 新增/刪除表格資料

- alter table `name` add `attri_name` data_type;
- alter table `name` drop column `attri_name`;
- 不論新增或刪除，都是以alter為開頭



新增



刪除

Chap. 4 儲存資料（insert）

建功高中物理科
cow3690m@tmail.hc.edu.tw

呂昀修

Chap. 4 儲存資料（insert）

- 儲存資料：
  - insert into `table_name` value(col1_data, col2_data, …, coln_data);

```
insert into `employee` values(206, 'Alice', '1999/10/8', 'F', 50000);
```

```
47  18:07:36  insert into `employee` values(206, 'Alice', '1999/10/8', 'F', 50000)
```

- 顯示資料：
  - select * from `table_name`;
  - 「*」表示所有欄位；若指定某欄位，則只

```
22 •  select * from `employee`;
```

Result Grid | Filter Rows: | Edit:

employee_id	name	birthday	gender	salary
206	Alice	1999/10/8	F	50000
NULL	NULL	NULL	NULL	NULL

Chap. 4 儲存資料（insert）

- 儲存資料：
  - 也可暫時不填入資料：

```
21 • insert into `employee` values(207, 'Bob', NULL, 'M', 30000);
22 • select * from `employee`;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

employee_id	name	birthday	gender	salary
206	Alice	1999/10/8	F	50000
207	Bob	NULL	M	30000
NULL	NULL	NULL	NULL	NULL

  - 但要補入資料就必須用update，請翻到後面查閱

Chap. 4-1 限制 / 約束資料

建功高中物理科
cow3690m@tmail.hc.edu.tw

呂昀修

Chap. 4-1 限制 / 約束資料

- 先將原本的表格刪除（drop table `table_name`）
- 我們可以對表格中的資料加上一些限制：

```
create table `employee` (
    `employee_id` int primary key,
    `name` varchar(20) not null,
    `birthday` varchar(20) unique,
    `gender` varchar(20),
    `salary` int
);
```

如原本的primary key寫在這邊，就是一種限制

限制：在插入name資料時，不可為null

限制：birthday必須為唯一值

Chap. 4-1 限制 / 約束資料

- 先將原本的表格刪除（drop table `table_name`）
- 我們可以對表格中的資料加上一些限制：

```
create table `employee` (
    `employee_id` int primary key,
    `name` varchar(20) not null,
    `birthday` varchar(20) unique,
    `gender` varchar(20),
    `salary` int
);
```

- 創建後describe發現：

Field	Type	Null	Key	Default	Extra
employee_id	int	NO	PRI	NULL	
name	varchar(20)	NO		NULL	
birthday	varchar(20)	YES	UNI	NULL	
gender	varchar(20)	YES		NULL	
salary	int	YES		NULL	

Chap. 4-1 限制 / 約束資料

- 先將原本的表格刪除（drop table `table_name`）
- 我們可以對表格中的資料加上一些限制：

```
create table `employee` (
    `employee_id` int primary key,
    `name` varchar(20) not null,
    `birthday` varchar(20) unique,
    `gender` varchar(20),
    `salary` int
);
```

- 創建後describe發現：

Field	Type	Null	Key	Default	Extra
employee_id	int	NO	PRI	NULL	
name	varchar(20)	NO		NULL	
birthday	varchar(20)	YES	UNI	NULL	
gender	varchar(20)	YES		NULL	
salary	int	YES		NULL	

- 我們對它新增資料看看：（假設我故意輸入null name）
  - 會發現出錯了

```
insert into `employee` values(206, null, '1999/10/8', 'F', null);
```

55 18:23:21 insert into `employee` values(206, null, 1999/10/8, 'F', null)

Chap. 4-1 限制 / 約束資料

- 先將原本的表格刪除（drop table `table_name`）
- 我們可以對表格中的資料加上一些限制：

```
create table `employee` (
    `employee_id` int primary key,
    `name` varchar(20) not null,
    `birthday` varchar(20) unique,
    `gender` varchar(20),
    `salary` int
);
```

- 創建後describe發現：

Field	Type	Null	Key	Default	Extra
employee_id	int	NO	PRI	NULL	
name	varchar(20)	NO		NULL	
birthday	varchar(20)	YES	UNI	NULL	
gender	varchar(20)	YES		NULL	
salary	int	YES		NULL	

- 我們對它新增資料看看：（假設我故意輸入null name）
- 同理，輸入相同的生日：

```
insert into `employee` values(207, 'Bob', '1999/10/8', 'M', null);
```

58 18:27:38 insert into `employee` values(207, 'Bob', '1999/10/8', 'M', null)

- 如果primary key資料是連續遞增 / 遞減，也可以使用：

```
create table `employee` (
    `employee_id` int primary key auto_increment,
    `name` varchar(20) not null,
    `birthday` varchar(20),
    `gender` varchar(20),
    `salary` int
);
```

- 這樣在新增資料時，就可以改成：

```
insert into `employee` values(206, 'Alice', '1999/10/8', 'F', 50000);
insert into `employee`(`name`, `birthday`, `gender`, `salary`) values('Bob', '1985/9/16', 'M', 30000);
insert into `employee`(`name`, `birthday`, `gender`, `salary`) values('Herry', '2000/12/19', 'M', 35000);
```

employee_id	name	birthday	gender	salary
206	Alice	1999/10/8	F	50000
207	Bob	1985/9/16	M	30000
208	Herry	2000/12/19	M	35000
NULL	NULL	NULL	NULL	NULL

Chap. 5 修改、刪除資料（update / delete）

建功高中物理科
cow3690m@tmail.hc.edu.tw
呂昀修

```
update `table_name`
set `colm_name`=value
where `coln_name`=value;
```
條件式

我們先新創一個table資料：

```
create table `branch` (
    `branch_id` int primary key auto_increment,
    `branch_name` varchar(20)
);
```

並且插入一些資料：

```
insert into `branch` values(1, 'RD');
insert into `branch`(`branch_name`) values('ADM');
insert into `branch`(`branch_name`) values('IT');
insert into `branch`(`branch_name`) values('EE');
```

	branch_id	branch_name
▶	1	RD
	2	ADM
	3	IT
	4	EE
*	NULL	NULL

```
update `table_name`
set `colm_name`=value
where `coln_name`=value;
```
條件式

	branch_id	branch_name
▶	1	RD
	2	ADM
	3	IT
	4	EE
*	NULL	NULL

情境：
部門整合，IT跟EE合併，變為IT/EE部門，但branch_id不變

```
update `branch`
set `branch_name` = 'IT/EE'
where `branch_name` = 'IT' or 'branch_name' = 'EE';
```

Chap. 5 修改資料

```
update `table_name`
set `colm_name`=value
where `coln_name`=value;
```
條件式

	branch_id	branch_name
▶	1	RD
	2	ADM
	3	IT
	4	EE
*	NULL	NULL

情境：
部門整合，IT跟EE合併，變為IT/EE部門，但branch_id不變

```
update `branch`
set `branch_name` = 'IT/EE'
where `branch_name` = 'IT' or 'branch_name' = 'EE';
```

執行下去你會發現：

```
95  18:55:51  update `branch` set `branch_name` = 'IE' where `branch_name` = 'IT' or `branch_name` = 'EE'
```

也就是說，一般的資料庫當然不允許你隨意進入更改！

所以我們必須加上：

```
set SQL_SAFE_UPDATES=0;
```

此為固定的環境變數，不可更改名字

	branch_id	branch_name
▶	1	RD
	2	ADM
	3	IE
	4	IE
*	NULL	NULL

Chap. 5 修改資料

```
update `table_name`
set `colm_name`=value
where `coln_name`=value;
```
條件式

	branch_id	branch_name
▶	1	RD
	2	ADM
	3	IT
	4	EE
*	NULL	NULL

情境：
部門整合，IT跟EE合併，變為IT/EE部門，但branch_id不變

```
update `branch`
set `branch_name` = 'IT/EE'
where `branch_name` = 'IT' or 'branch_name' = 'EE';
```

更新後再執行一次update便發現：

	branch_id	branch_name
▶	1	RD
	2	ADM
	3	IE
	4	IE
*	NULL	NULL

```
update `table_name`
set `colm_name`=value
where `coln_name`=value;
```

條件式

	branch_id	branch_name
▶	1	RD
	2	ADM
	3	IT
	4	EE
*	NULL	NULL

情境：
部門整合，IT跟EE合併，變為IT/EE部門，但branch_id不變

```
update `branch`
set `branch_name` = 'IT/EE'
where `branch_name` = 'IT' or 'branch_name' = 'EE';
```

這邊也可以改成：
where `branch_name` in('IT', 'EE');

更新後再執行一次update便發現：

	branch_id	branch_name
▶	1	RD
	2	ADM
	3	IE
	4	IE
*	NULL	NULL

Chap. 5 刪除資料

```
delete from `table_name`
where `coln_name`=value;
```
條件式

branch_id	branch_name
1	RD
2	ADM
3	IE
4	IE
5	HRRRR
NULL	NULL

情境：
新增一個HR部門，但新增資料時打成'HRRRR'，我們想要刪掉重來

```
delete from `branch`
where `branch_id` = 5;
```

```
40 •  delete from `branch`
41    where `branch_id` = 5;
42
43 •  select * from `branch`;
```

Result Grid | Filter Rows:

branch_id	branch_name
1	RD
2	ADM
3	IE
4	IE
NULL	NULL

Chap. 5 刪除資料

```
delete from `table_name`
where `coln_name`=value;
```
條件式

	branch_id	branch_name
▶	1	RD
	2	ADM
	3	IT
	4	EE
	5	HR
*	NULL	NULL

情境：
我們要把branch_id > 2的部門都刪除

```
delete from `branch`
where `branch_id` > 2
```

	branch_id	branch_name
▶	1	RD
	2	ADM
*	NULL	NULL

Chap. 6 取得資料
（select `col_name` from `table_name`）

建功高中物理科
cow3690m@tmail.hc.edu.tw
呂昀修

Chap. 6 取得資料（select `col_name` from `table_name`）

- 繼續用剛剛的employee做例子：
  - 右圖為select * from `employee`的結果
  - 若改成select `name` from `employee`
  - 若改成select `name`, `gender` from `employee`

employee_id	name	birthday	gender	salary
206	Alice	1999/10/8	F	50000
207	Bob	1985/9/16	M	30000
208	Herry	2000/12/19	M	35000
209	Eve	1997/1/229	F	47000
210	John	2000/12/19	M	68000
NULL	NULL	NULL	NULL	NULL

name
Alice
Bob
Herry
Eve
John

name	gender
Alice	F
Bob	M
Herry	M
Eve	F
John	M

- 繼續用剛剛的employee做例子：

employee_id	name	birthday	gender	salary
206	Alice	1999/10/8	F	50000
207	Bob	1985/9/16	M	30000
208	Herry	2000/12/19	M	35000
209	Eve	1997/1/229	F	47000
210	John	2000/12/19	M	68000
NULL	NULL	NULL	NULL	NULL

  - 亦可加上顯示資料的排序方式：
  - select * from `employee` order by `salary` desc;

employee_id	name	birthday	gender	salary
210	John	2000/12/19	M	68000
206	Alice	1999/10/8	F	50000
209	Eve	1997/1/229	F	47000
208	Herry	2000/12/19	M	35000
207	Bob	1985/9/16	M	30000
NULL	NULL	NULL	NULL	NULL

  - 升序則是： select * from `employee` order by `salary`;

- 繼續用剛剛的employee做例子：
  - 排序後只取兩筆數據：
  - select * from `employee` order by `salary` desc limit 2;

employee_id	name	birthday	gender	salary
206	Alice	1999/10/8	F	50000
207	Bob	1985/9/16	M	30000
208	Herry	2000/12/19	M	35000
209	Eve	1997/1/229	F	47000
210	John	2000/12/19	M	68000
NULL	NULL	NULL	NULL	NULL

employee_id	name	birthday	gender	salary
210	John	2000/12/19	M	68000
206	Alice	1999/10/8	F	50000
NULL	NULL	NULL	NULL	NULL