

MySQL 官方網站

<http://dev.mysql.com>

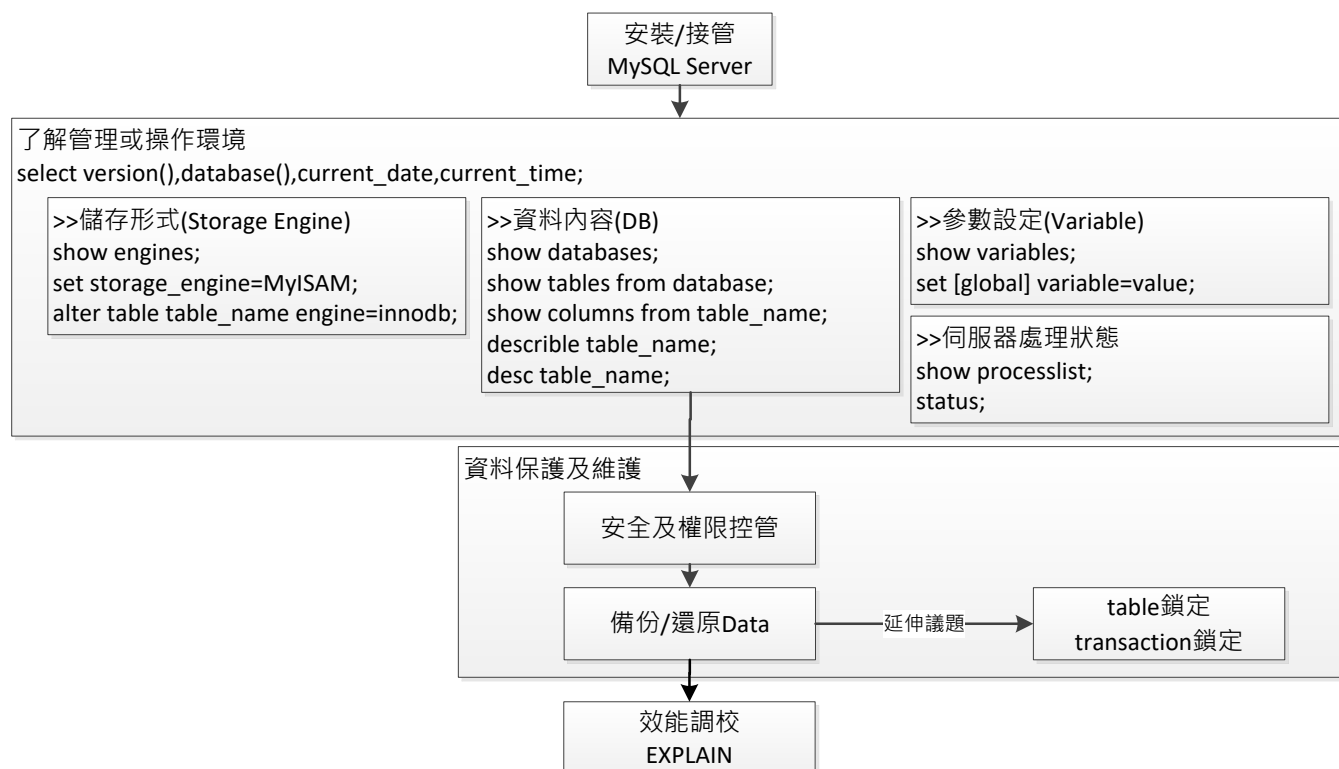
MySQL Client 管理工具

1. mysql 用戶端程式 (內建 command line 工具)
2. MySQL workbench **[推薦]**
3. 跨平台且跨資料庫的開源用戶端程式 – SquirrelL
4. SQLyog **[推薦]**

<https://github.com/webyog/sqlyog-community/wiki/Downloads>

5. phpMyAdmin **[不推，有資安漏洞]**

MySQL 維護管理邏輯思路



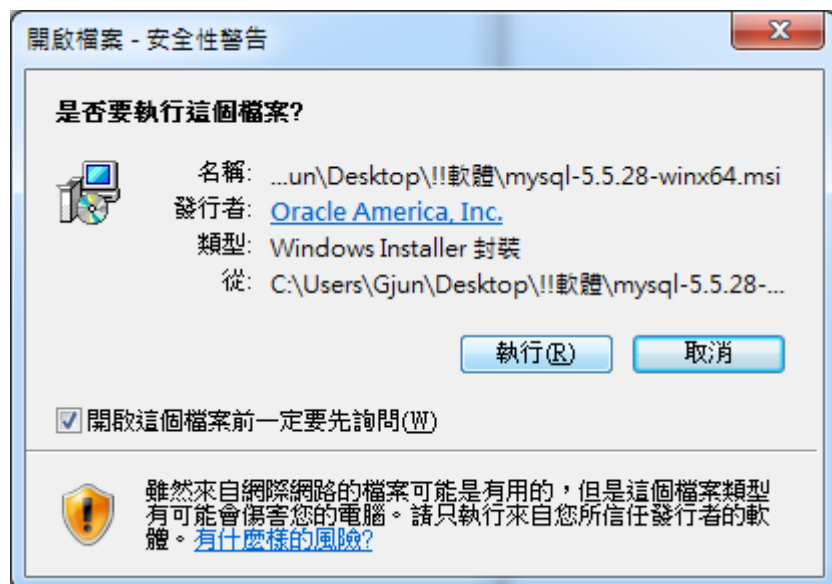
MySQL 安裝

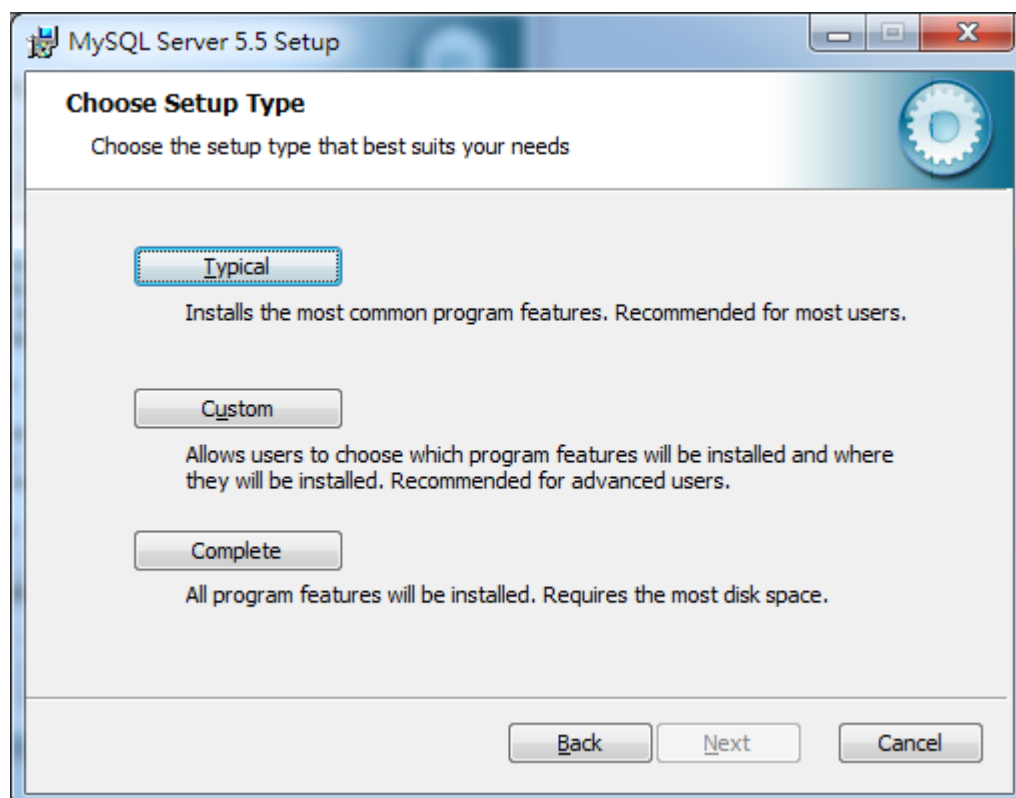
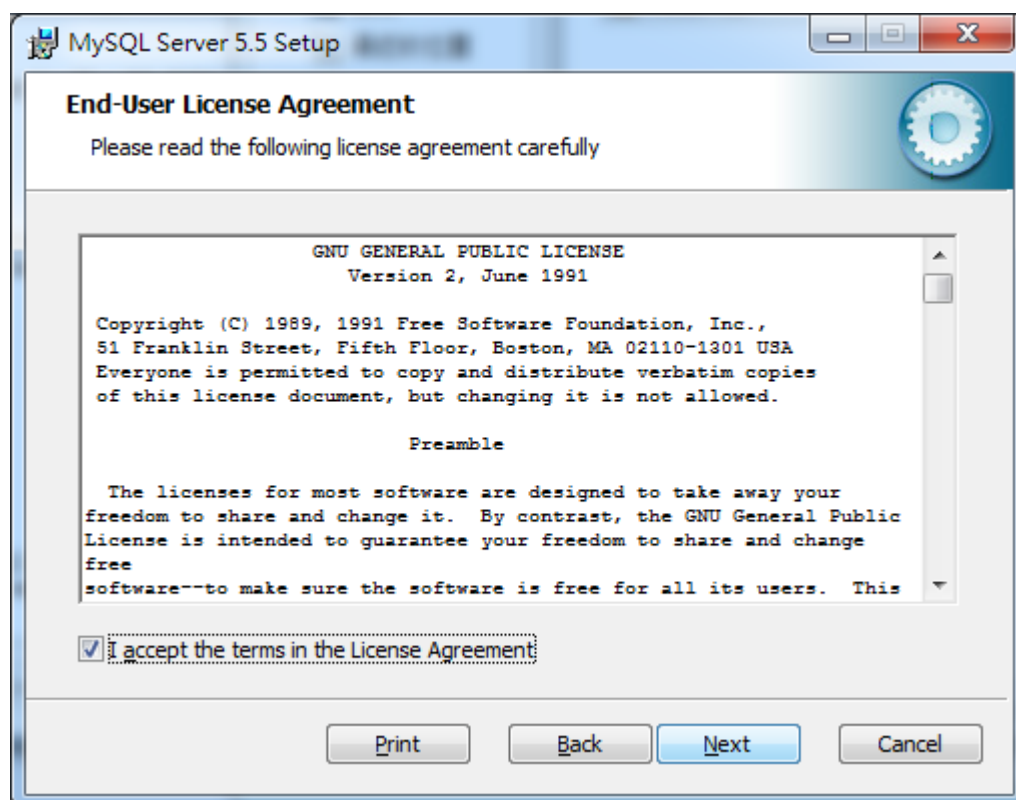
Windows Option :

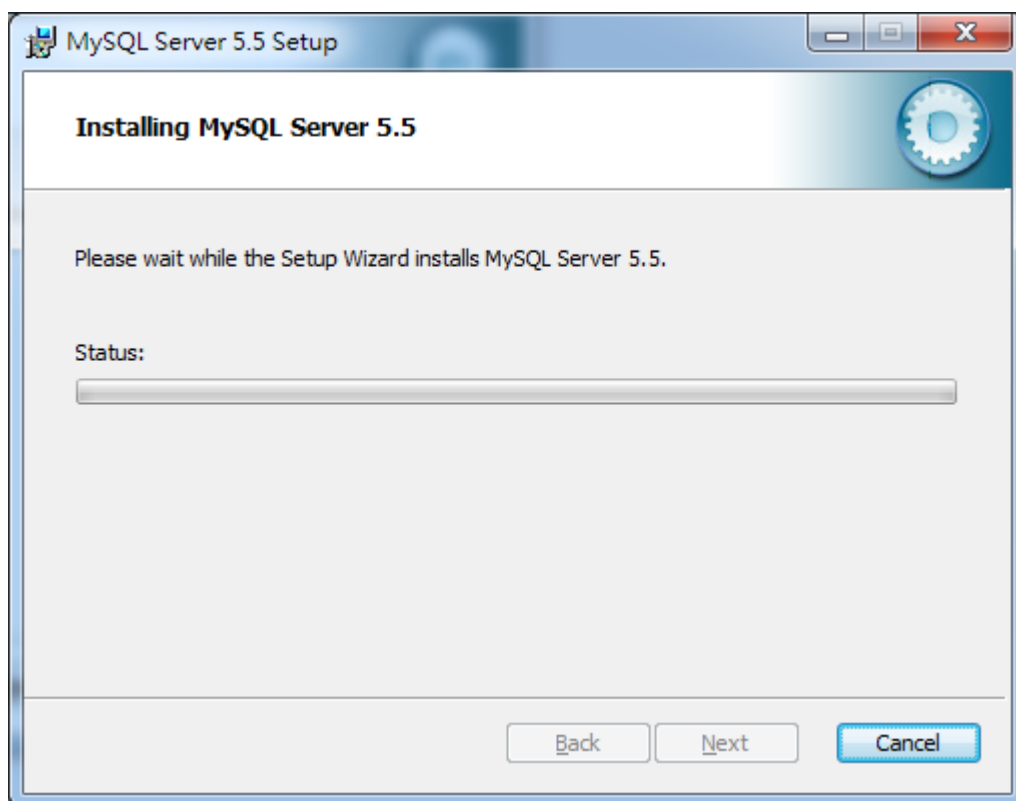
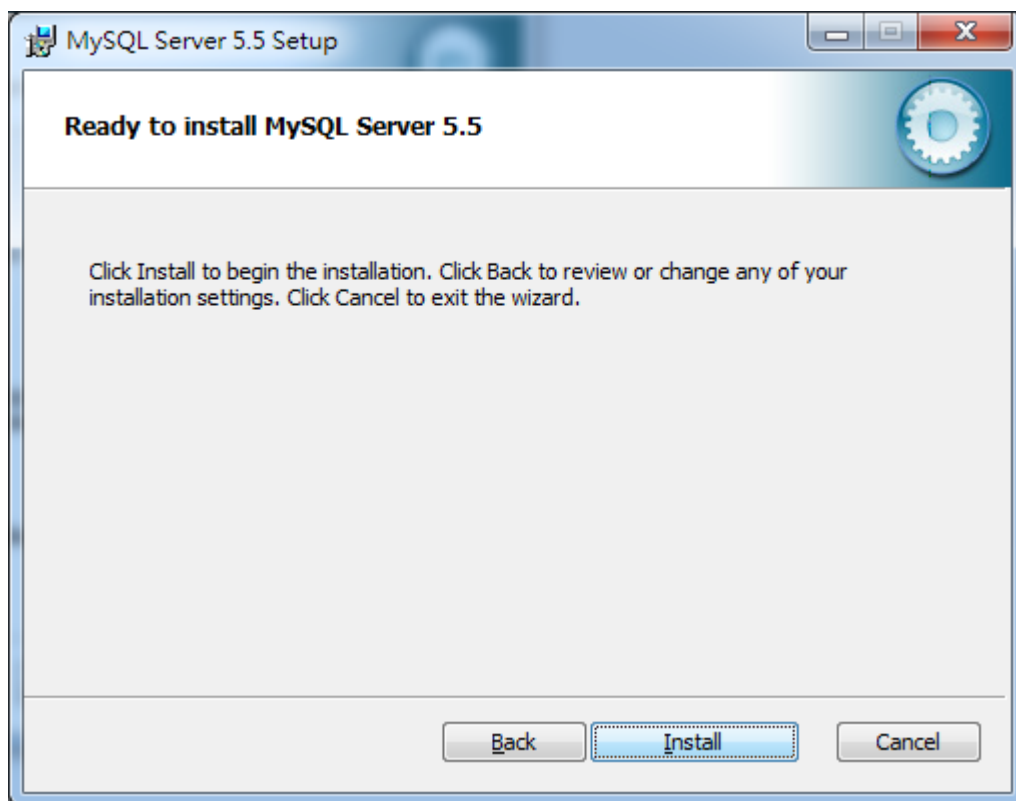
勾選 Include Bin Directory in Windows Path

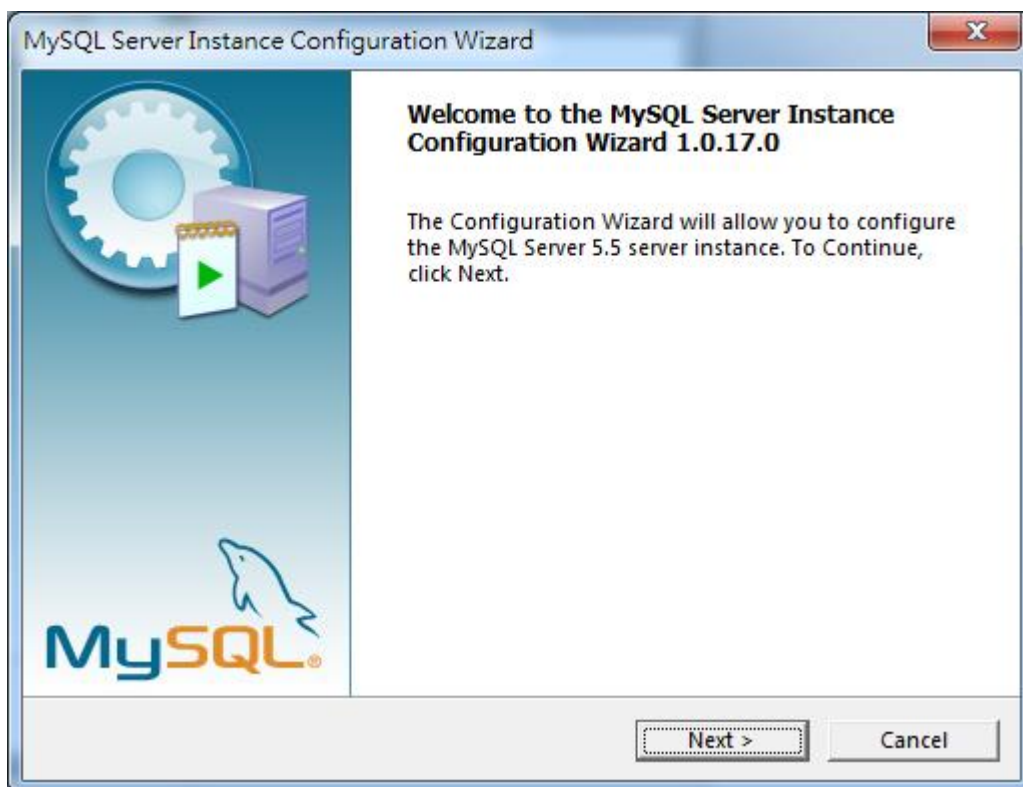
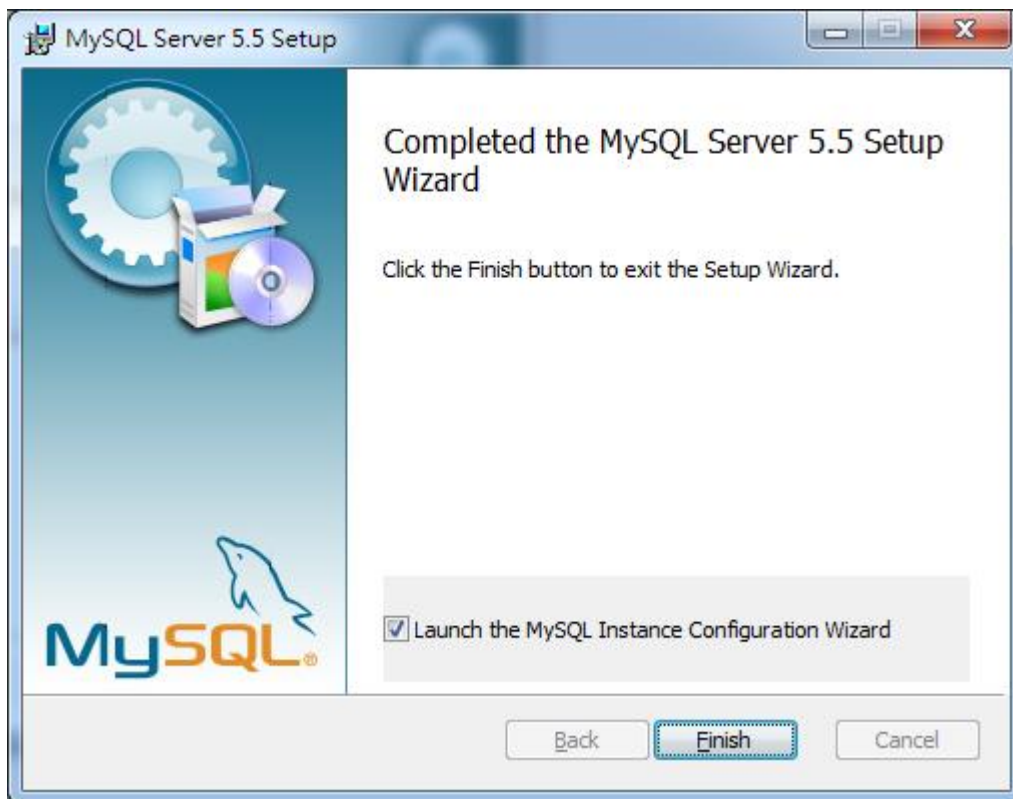
Security Option :

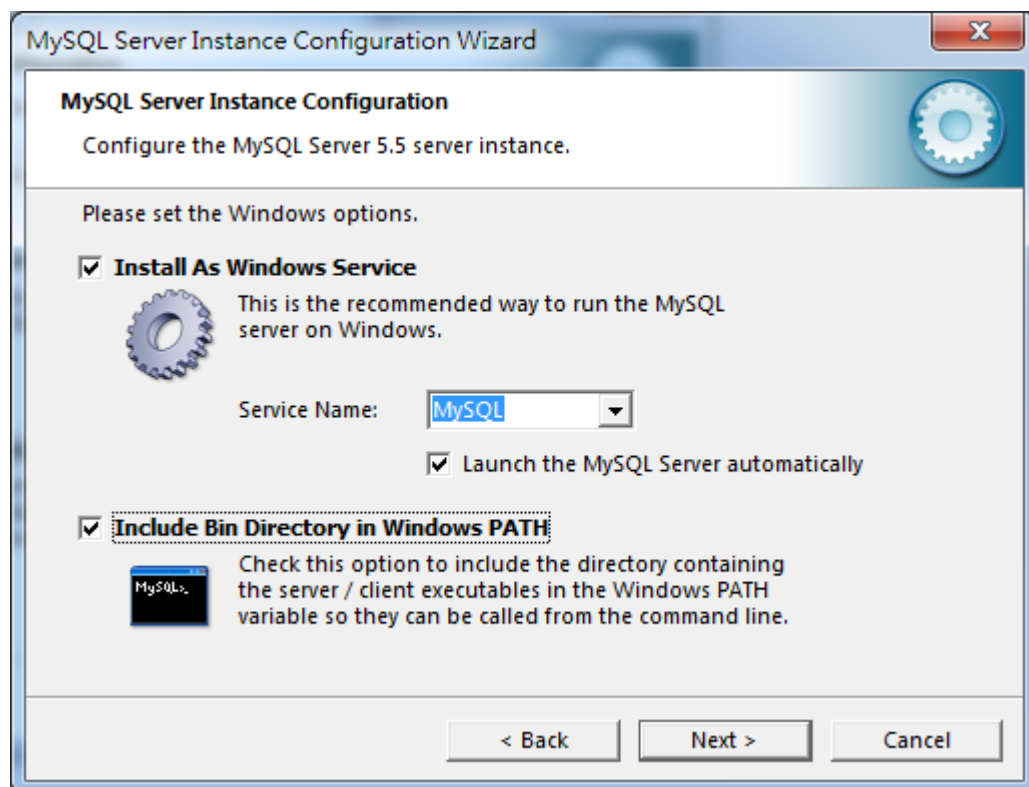
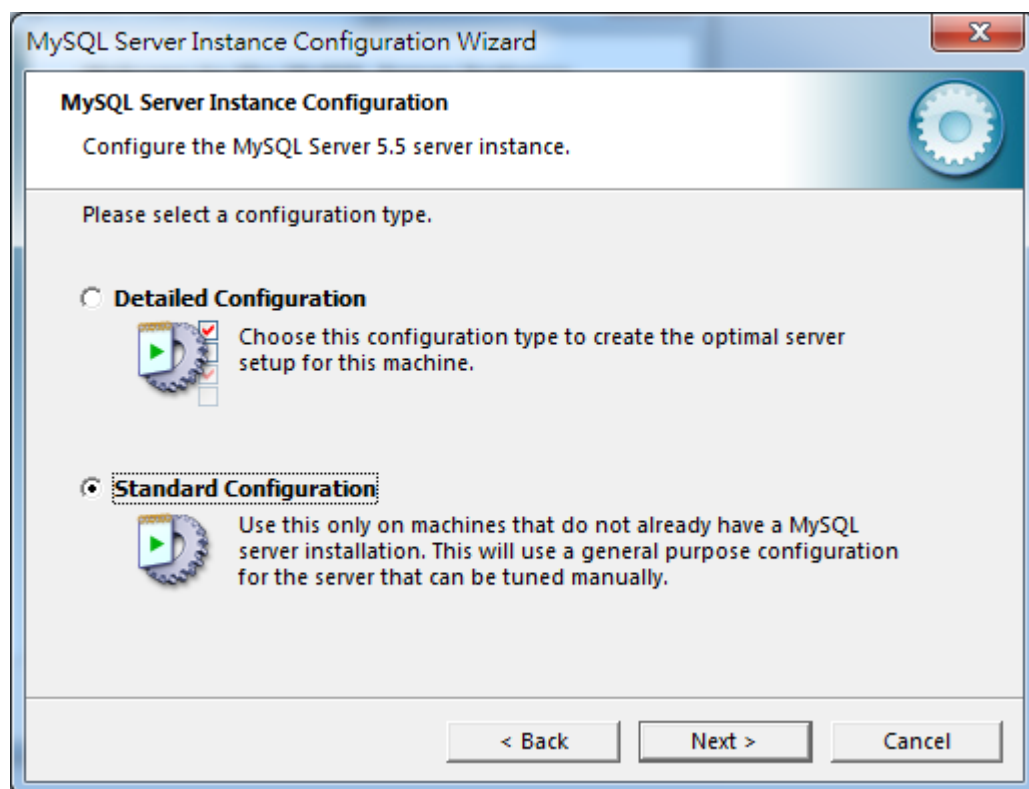
勾選 Enable Root access from remote machines

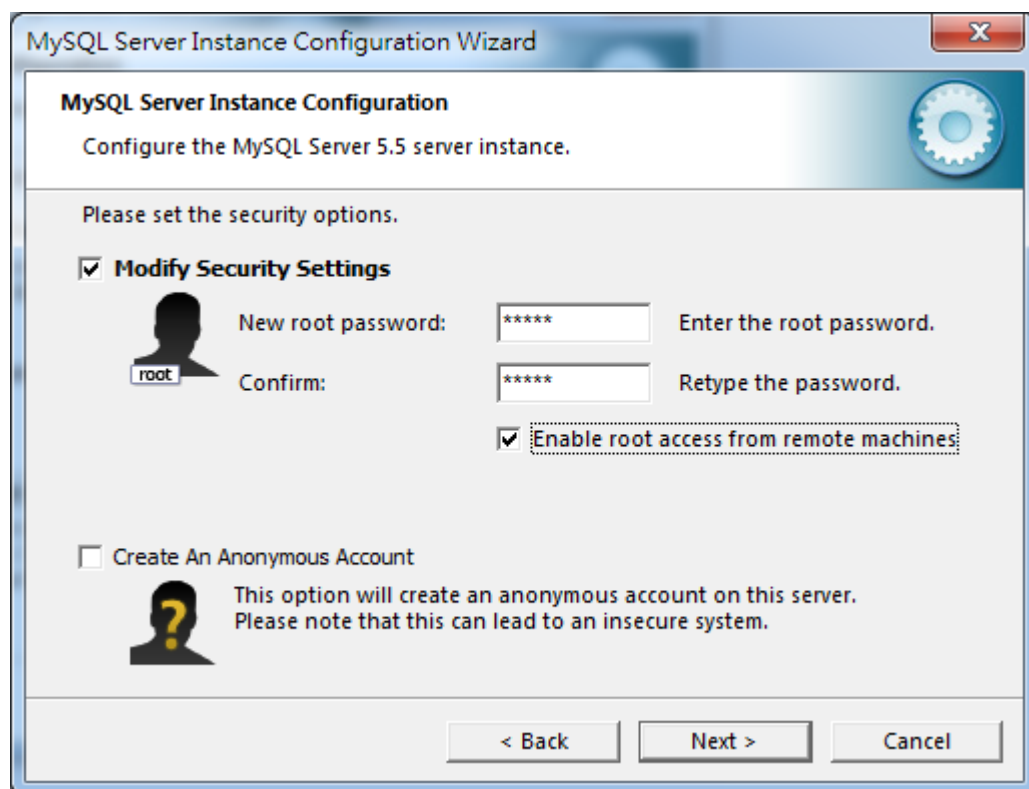




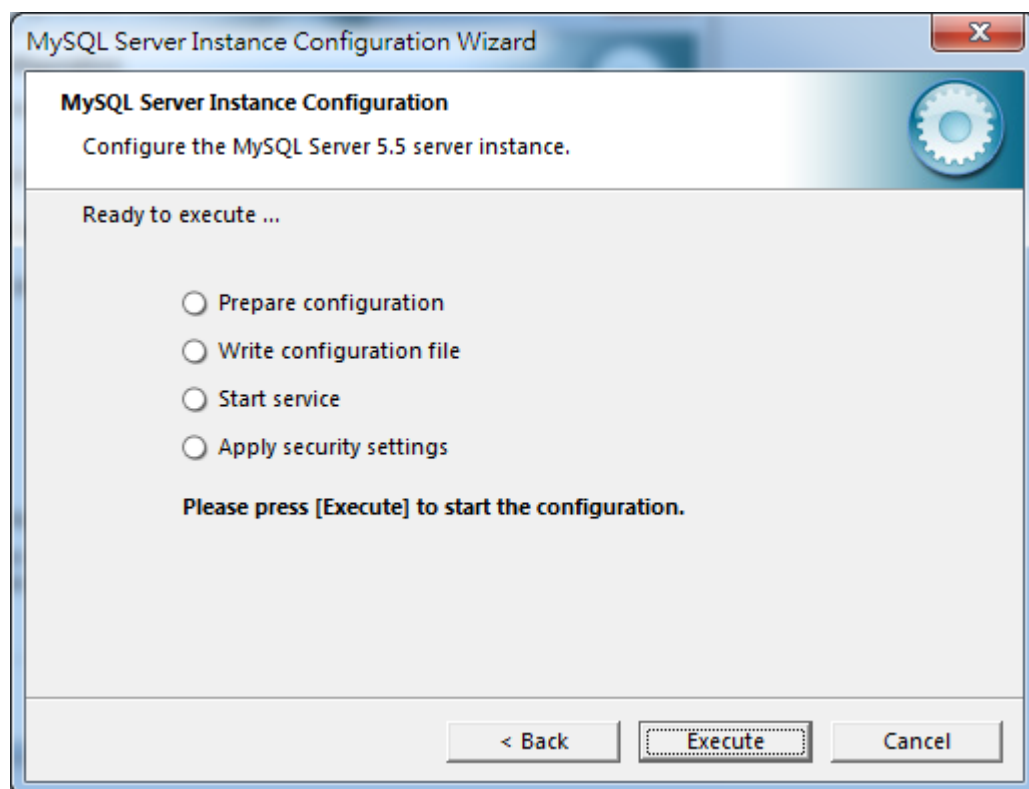


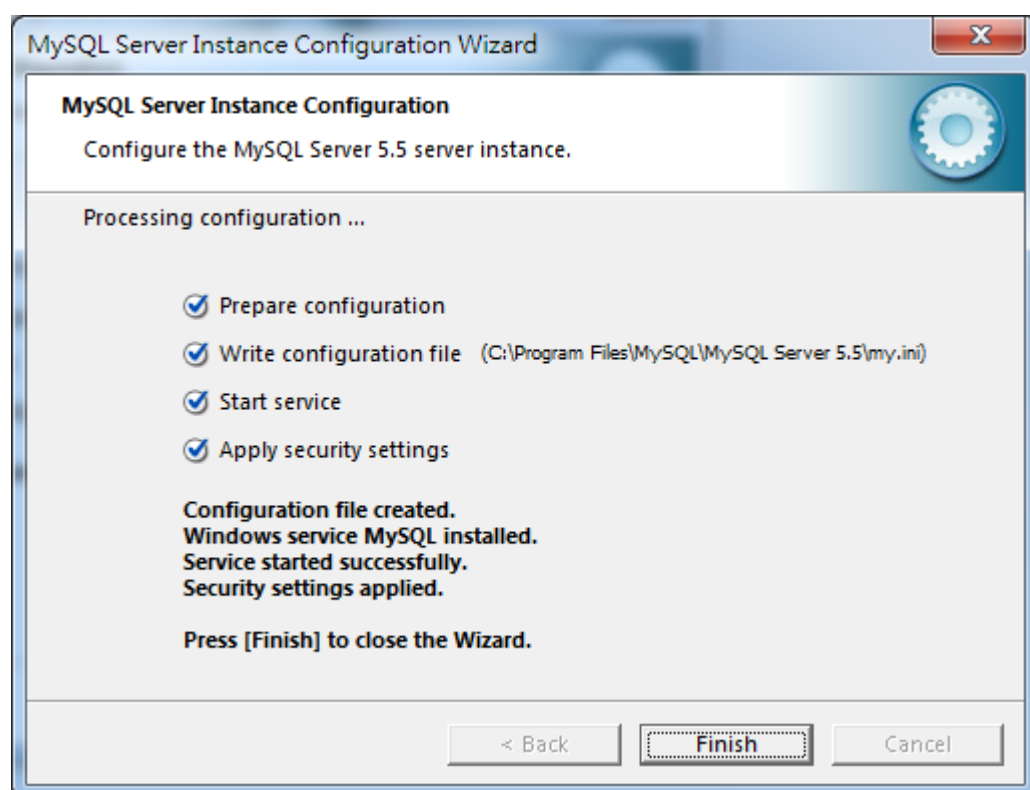




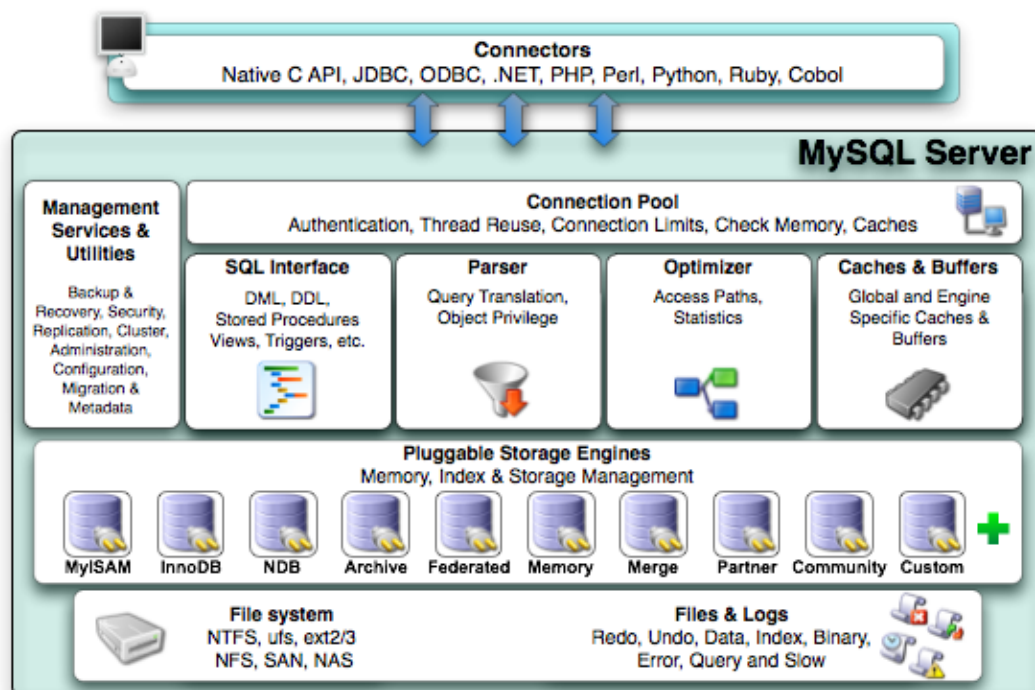


※Root 密碼設定→不要用「@」、「\」、「:」、「,」等字眼。





MySQL 架構



MySQL 儲存引擎(Storage Engines)

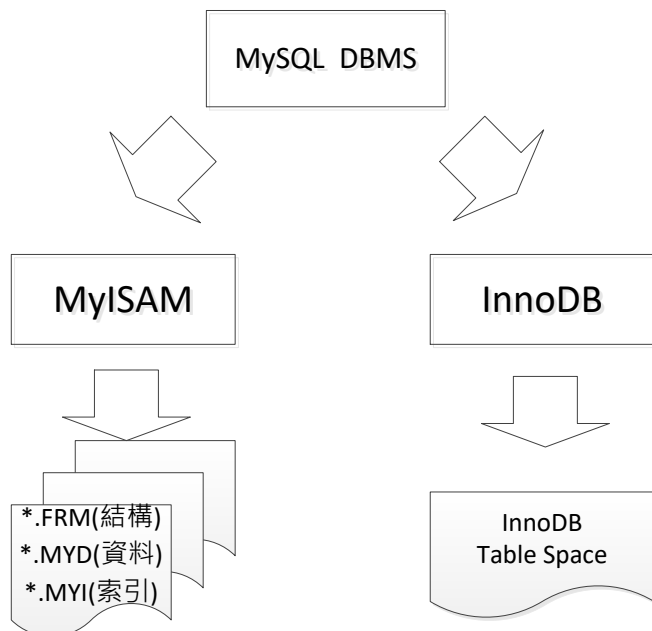
Feature	MyISAM	Memory	InnoDB	Archive	NDB
Storage limits	256TB	RAM	64TB	None	384EB
Transactions	No	No	Yes	No	Yes
Locking granularity	Table	Table	Row	Table	Row
MVCC	No	No	Yes	No	No
Geospatial data type support	Yes	No	Yes	Yes	Yes
Geospatial indexing support	Yes	No	No	No	No
B-tree indexes	Yes	Yes	Yes	No	Yes
Hash indexes	No	Yes	No ^[a]	No	Yes
Full-text search indexes	Yes	No	No	No	No
Clustered indexes	No	No	Yes	No	No
Data caches	No	N/A	Yes	No	Yes
Index caches	Yes	N/A	Yes	No	Yes
Compressed data	Yes ^[b]	No	Yes ^[c]	Yes	No
Encrypted data ^[d]	Yes	Yes	Yes	Yes	Yes
Cluster database support	No	No	No	No	Yes
Replication support ^[e]	Yes	Yes	Yes	Yes	Yes
Foreign key support	No	No	Yes	No	No
Backup / point-in-time recovery ^[f]	Yes	Yes	Yes	Yes	Yes
Query cache support	Yes	Yes	Yes	Yes	Yes
Update statistics for data dictionary	Yes	Yes	Yes	Yes	Yes

※兩大主流→InnoDB、MyISAM

分水嶺：MySQL5.5

之前是以 MyISAM 為預設的儲存引擎。

之後(含)是以 InnoDB 為預設的儲存引擎。



MyISAM 主要特色	InnoDB 主要特色
(1)支援全文檢索。	(1)支援交易、ACID。
(2)支援 table-level 鎖定。	(2)支援外來鍵。
	(3)支援 row-level 鎖定。

<Remark> 使用 MyISAM 注意事項

1. 因為 tables 最終是會壞掉的 (是沒錯)，所以要記得備份，單看重要性吧，每天、或每週，或者用 mysql replication 將資料傳到另一台主機
2. 在 my.cnf 加入設定讓它有自動修復的功能
myisam-recover=backup,force
3. 或者寫程式排程固定時間檢查 table 的狀態
<http://www.softwareprojects.com/resources/programming/t-mysql-table-maintenance-automation-1400.html>

[REF] <http://www.softwareprojects.com/resources/programming/t-mysql-storage-engines-1470.html>

※各種 storage engines 的特色與評選

[官方文件] <https://dev.mysql.com/doc/refman/5.7/en/storage-engines.html>

MyISAM：最常用的，但不支援交易管理功能。

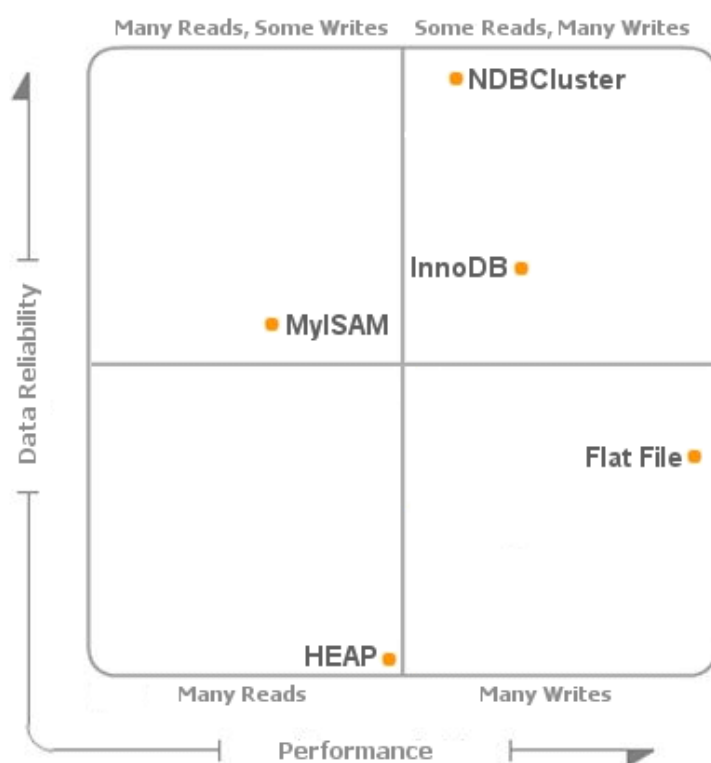
InnoDB：適合需要大量資料寫入的環境，支援交易管理功能。

MERGE：合併 MyISAM 的相同結構的 table，提供單一預覽!!!?

Federated：影子資料表。

HEAP(Memory)：資料存在記憶體中，最適合給暫存用的 table。

NDB：叢集環境。



※[實作]：查詢 mysql 支援的 storage engine

```
mysql> show engines\G
```

※[實作]：更改資料表所使用的 storage engine

```
mysql> alter table table_name engine=innodb;
```

※[實作]：更改預設使用的 storage engine

```
mysql> set storage_engine=MyISAM;
```

<Remark> MyISAM

1. 每個 MyISAM 的資料表，在資料庫資料夾都各有以下三個檔案：
 - A. 資料表名稱.FRM （結構檔）
 - B. 資料表名稱.MYD （資料檔）
 - C. 資料表名稱.MYI （索引檔）
2. 不支援 Transaction，也沒有外鍵 Constraint。
3. 具有最富彈性的 AUTO_INCREMENT 功能。
4. MyISAM 資料表可組合成 MERGE 資料表。
5. MyISAM 資料表可轉成唯讀+壓縮的封存模式。
6. MyISAM 支援 FULLTEXT 全文檢索功能。
7. MyISAM 採用 table-level 鎖定。

<Remark> InnoDB

1. InnoDB 的資料表，在資料庫資料夾只有 .FRM 結構檔。
2. 所有的 InnoDB 的資料表，資料統一放在 InnoDB 的 Table Space。
3. 支援完整的 Transaction ACID 功能。
4. 支援外鍵 Constraint。
5. 自動資料修復。
6. InnoDB 採用 row-level 鎖定。

※[實作] 比較 MyISAM 和 InnoDB 之間的差異

```
use test;

drop table Lab;

create table Lab
(
    id int auto_increment primary key,
    data int not null
) engine = MyISAM;
-- ) engine = InnoDB;

insert into Lab (data) values (100);

select * from Lab;

start transaction;

update lab set data = 200 where id = 1;

-- rollback transaction;
rollback;

select * from Lab;
```

<Remark> Memory (HEAP)

1. Memory 的資料表，在資料庫資料夾只有 .FRM 結構檔。
2. 資料放在記憶體，停電或 MySQL 重新啟動，資料將自動消失。
3. 適合階段性的暫存資料。

<Remark> Merge

1. 聯合多個 MyISAM 的資料表，組合成一個邏輯資料表。
2. Merge 型的資料表，在資料庫資料夾會有 .FRM 、.MRG 檔案。
3. 實際上的資料放在各個 MyISAM 的資料表。

※實作

```
use test;

drop table Lab;
drop table t1;
drop table t2;

create table t1
(
    id int auto_increment primary key,
    data varchar(5) not null
) engine = MyISAM;

create table t2
(
    id int auto_increment primary key,
    data varchar(5) not null
) engine = MyISAM;

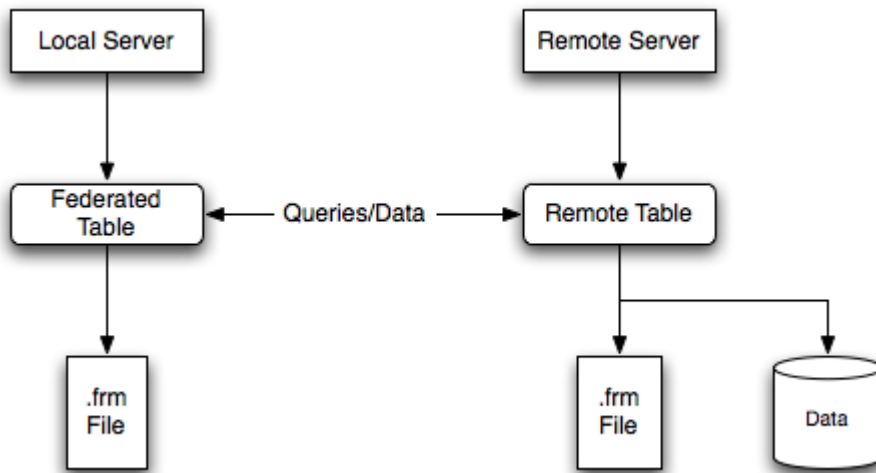
create table Lab
(
    id int auto_increment unique,
    data varchar(5) not null
) engine = Merge union = (t1, t2) insert_method = first;
-- insert_method 選項: no | first | last

insert into t1 (data) values ('t1');
insert into t2 (data) values ('t2');
insert into Lab (data) values ('rec3');
insert into Lab (data) values ('rec4');

select * from Lab;
select * from t1;
select * from t2;
```

<Remark> Federated

從 mysql5.0.3 開始內建支援 federated，目的是讓 local 的資料庫可能與遠端的資料庫協同合作，使用者不必將資料複製一份到本地資料庫，也可以存取遠端的資料。(類似 Oracle 的 DB Link)



1. 類似一個影子資料表。
2. 實際上的資料置於本尊資料表，本尊資料表置於本機或遠端皆可。
3. 使用者或應用程式可透過影子資料表，間接存取本尊資料表。
4. 建立 Federated 資料表的語法：

```
CREATE TABLE 影子資料表名稱 (
    各欄位名稱 型態      --必須與本尊資料表相同
) engine = federated connection =
'mysql://user:password@remote_host:port/dbname/本尊資料表';
```

※[實作]：Federated

Step 1 查看 mysql 是否支援 Federated (SQL 指令：show engines\G)

Step 2 在設定檔(my.ini/my.cnf)中[mysqld]的下方加入「federated」字眼。

Step 3 重跑 mysql 服務。

Win 平台下：

```
shell> net stop mysql
```

```
shell> net start mysql
```

CentOS/Fedora Linux 平台下：

```
shell> service mysql restart → CentOS 6.X 以前
```

```
shell> systemctl start mysqld → CentOS 7.X 以後
```

Step 4 再一次查看 mysql 是否支援 Federated (SQL 指令：show engines\G)

Step 5 建立本尊表 test_table

Step 6 建立分身表 shadow_table

Step 7 在分身表 shadow_table 插入 data

Step 8 查詢本尊表 test_table 的 data 內容

```
-- 使用 test DB
use test;

-- 先刪存在的表
drop table if exists test_table;
drop table if exists shadow_table;

-- 建立本尊表
create table test_table ( id int, data int );

-- 建立分身表
create table shadow_table ( id int, data int ) engine=federated
connection='mysql://root:psps3773@localhost:3306/test/test_table';

-- 查看目前資料表
show tables from test;
show table status from test \G

-- 針對分身表插入 data
insert into shadow_table (id,data) values (1,20);

-- 查看資料
select * from test_table;

-- 查看目前資料表狀態
show table status from test \G
```

MySQL 命令列模式簡介

※登入 mysql 命令列模式

```
shell> mysql -u 帳號 -p
```

※SQL 指令結尾用 ; 或 \G

(1)用分號(;)結尾者→table 觀點呈現

```
mysql> show variables like 'char%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| character_set_client | utf8 |
| character_set_connection | utf8 |
| character_set_database | utf8 |
| character_set_filesystem | binary |
| character_set_results | utf8 |
| character_set_server | utf8 |
| character_set_system | utf8 |
| character_sets_dir | /usr/share/mysql/charsets/ |
+-----+-----+
8 rows in set (0.00 sec)
```

(2)用\G 結尾者→row 觀點呈現

```
mysql> show variables like 'char%'\G
***** 1. row *****
Variable_name: character_set_client
Value: utf8
***** 2. row *****
Variable_name: character_set_connection
Value: utf8
***** 3. row *****
Variable_name: character_set_database
Value: utf8
***** 4. row *****
Variable_name: character_set_filesystem
Value: binary
***** 5. row *****
Variable_name: character_set_results
Value: utf8
***** 6. row *****
Variable_name: character_set_server
Value: utf8
***** 7. row *****
Variable_name: character_set_system
Value: utf8
***** 8. row *****
Variable_name: character_sets_dir
Value: /usr/share/mysql/charsets/
8 rows in set (0.00 sec)
```

※清除現有指令輸入內容 → \c

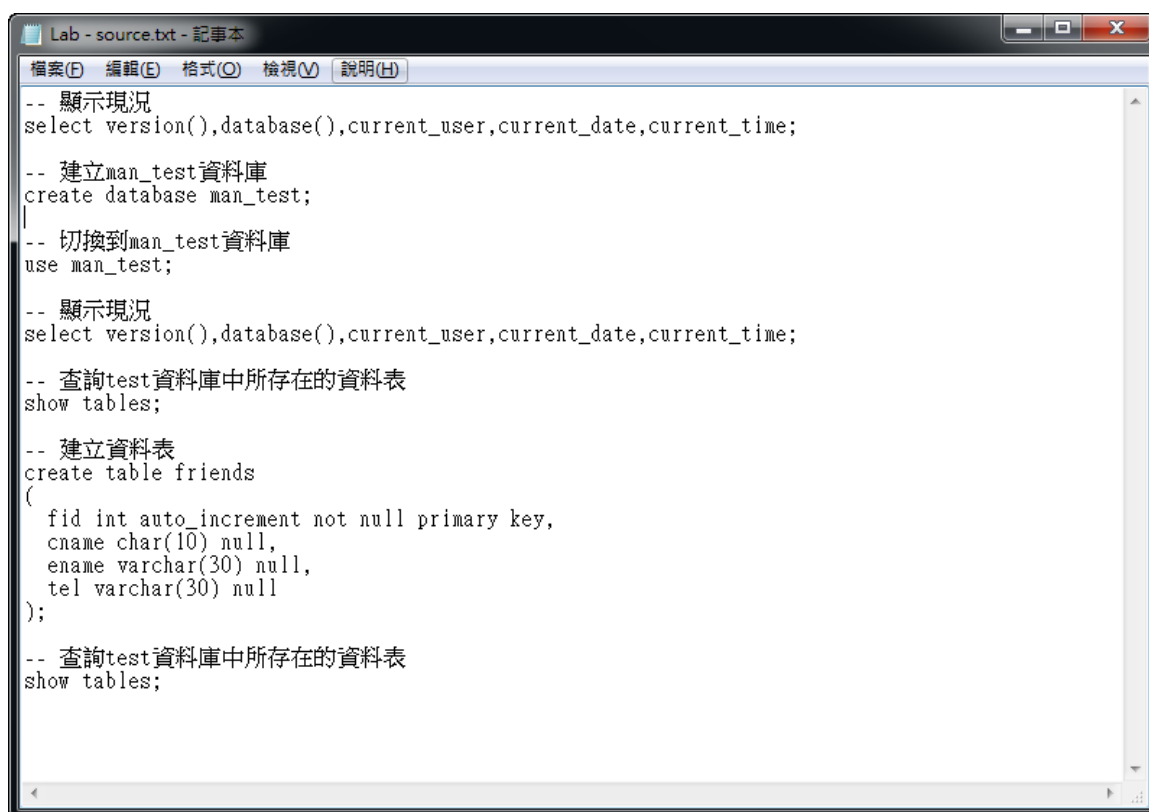
```
mysql> select * from
-> \c
mysql> █
```


※匯入 TXT 檔或 SQL 檔執行 SQL 指令

前提：

用 mysql 的指令模式編輯 sql 指令極為不便，建議先用相關文字檔編輯器編輯後儲存檔案，再利用 source 指令匯入文字檔執行之。

(1) 假設在 c:\有一個 source.txt 的 sql 指令文件檔如下：



```
-- 顯示現況
select version(),database(),current_user,current_date,current_time;

-- 建立man_test資料庫
create database man_test;

-- 切換到man_test資料庫
use man_test;

-- 顯示現況
select version(),database(),current_user,current_date,current_time;

-- 查詢test資料庫中所存在的資料表
show tables;

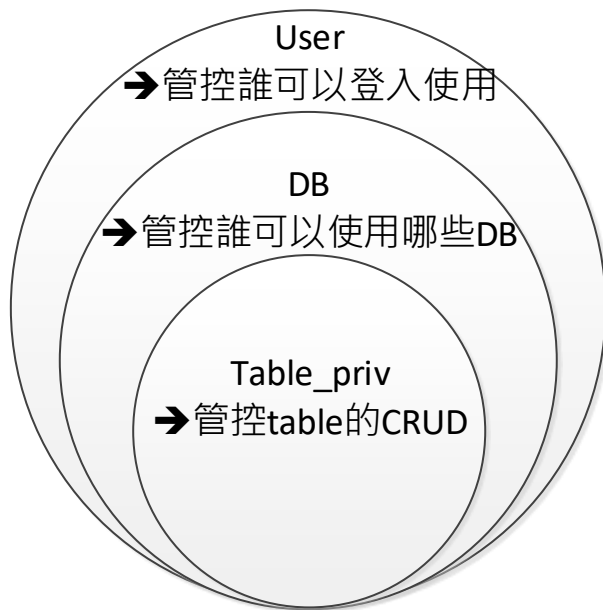
-- 建立資料表
create table friends
(
  fid int auto_increment not null primary key,
  cname char(10) null,
  ename varchar(30) null,
  tel varchar(30) null
);

-- 查詢test資料庫中所存在的資料表
show tables;
```

(2) 在 mysql 指令模式下鍵入 source c:\source.txt 或 source c:/source.txt 即可匯入執行指令。

MySQL 帳戶權限管理

※用三張資料表管控基本權限



※帳號管理：

1. 大、小寫有別，最長 16 字元。
2. 密碼設定 → 不要用「@」、「\」、「:」、「,」等字眼。
3. 權限管控標的 → 帳號@主機名稱。
4. 管控缺陷 → 沒有群組角色的觀念、沒有排除的觀念。

※mysql 資料庫用管控權限的資料表

user*
host (已廢棄不用)
db*
tables_priv*
columns_priv
procs_priv
proxies_priv

※[實作]：刪匿名帳號

```
mysql> use mysql;  
mysql> delete from user where user='';  
mysql> flush privileges;
```

※[實作]：修改使用者密碼

```
mysql> use mysql;  
mysql> update user set password=password('new password') where user='new user';  
mysql> flush privileges;
```

(註)：mysql5.7 之後的版本 password 欄位名稱改為 authentication_string

※[實作]：忘記 Root 密碼

1. 停止 mysql 服務。
2. 以略過管控表的方式啟動 mysql 服務。
`shell> mysqld -u root --skip-grant-tables &`
3. 另開 CMD 命令列進入 mysql
`shell> mysql`
4. 更新 root 密碼
`mysql> use mysql;`
`mysql> update user set password=password('new password') where user='root';`
`mysql> flush privileges;`
5. 查看工作管理員內「處理程序」是否仍有 mysqld 程式服務存在，存在者結束其作業，然後重新啟動 mysql 服務。

(註)：mysql5.7 之後的版本 password 欄位名稱改為 authentication_string

※[實作]：解決 client does not support authentication protocol requested by server

前提知識：

錯誤發生的原因是 4.1 版以後的密碼採用新的密碼驗證機制，如果用新版的 mysql client 工具去新增出來的使用者，會採用新的機制給予密碼，舊的 mysql client 工具連進來要驗證密碼時就會出現上述錯誤。

解法：

1. PHP5 改用 php_mysqli
2. 改用舊有演算法

將該要給舊 mysql client 工具使用的帳號改成舊有的密碼機制，例如用新的 mysql client 工具做以下動作。

```
mysql> use mysql;
mysql> SET PASSWORD FOR 'some_user'@'some_host' = OLD_PASSWORD('new_password');
mysql> FLUSH PRIVILEGES;
或
mysql> use mysql;
mysql> UPDATE user SET password=OLD_PASSWORD('new_password') WHERE user='some_user' and
host='some_host';
mysql> FLUSH PRIVILEGES;
```

(註)：mysql5.7 之後的版本 password 欄位名稱改為 authentication_string

<Remark> 限制連線數

整體設定

shell> vi /etc/my.cnf (直接修改 mysql 全域參數設定檔)

#mysql5

max_connections=100 #全部接受的最大連線數

max_user_connections=15 #每個使用者最大連線數

#mysql4

set-variable= max_connections=100 #全部接受的最大連線數

set-variable=max_user_connections=15 #每個使用者最大連線數

個別設定

➔針對各別使用者設定最大連線數 (這樣並不會變動到全域設定)

mysql> REVOKE ALL PRIVILEGES ON 資料庫 FROM 使用者@localhost;

mysql> REVOKE GRANT OPTION ON 資料庫 FROM 使用者@localhost;

mysql> GRANT 權限 ON 資料庫 TO 使用者 WITH MAX_USER_CONNECTIONS 15;

<Remark> 在 MySQL 中可以使用的註解(Comment)

1. #: "#"號開始的文字都是註解
2. --: "--"雙減號開始的文字也是註解
3. /*....*/: 用一對"/"及"*"括起來的文字是註解

MySQL 備份還原

※備份使用 mysqldump 工具

◎MySQL 單一資料庫備份還原語法：

備份

```
shell> mysqldump -u 使用者名稱 -p 密碼 資料庫名稱 > 備份檔名.sql
```

或

```
shell> mysqldump -u 使用者名稱 -p 密碼 資料庫名稱 > 備份檔名.sql --default-character-set=latin1
```

(中文資料庫有語系困擾時)

還原

```
shell> mysql -u 使用者名稱 -p 密碼 資料庫名稱 < 備份檔名.sql
```

◎MySQL 全部資料庫備份還原語法：

備份

```
shell> mysqldump -u 使用者名稱 -p 密碼 --all-database > 備份檔名.sql
```

或

```
shell> mysqldump -u 使用者名稱 -p 密碼 --all-database > 備份檔名.sql --default-character-set=latin1
```

(中文資料庫有語系困擾時)

還原

```
shell> mysql -u 使用者名稱 -p 密碼 < 備份檔名.sql
```

◎注意事項：

1. 備份檔內容之資料庫名或資料表名中若含'-'，則該名稱前後要加''，以免還原時出錯！使用舊版之 mysqldump 時會有所缺漏。
2. 新版之 MySQL 對於關鍵字有所增補，如：release……等。若舊版有使用到(資料表或欄位名稱)可能造成還原時出錯！

※MySQL 自動備份 Shell Script (For Linux)

→ <https://www.phpini.com/linux/mysql-backup-shell-script>

→ <https://blog.technologyofkevin.com/?p=389>

→ <http://blog.jangmt.com/2009/10/mysql-shell-script.html>

官方文件

<https://dev.mysql.com/doc/refman/5.7/en/mysqldump.html>

<https://mariadb.com/kb/en/mariadb/mysqldump/>

參考文件

<https://blog.roga.tw/2008/11/1127>

<http://jax-work-archive.blogspot.tw/2009/07/mysqldump-51-mysql.html>

<http://www.neo.com.tw/archives/1122>

交易管理與鎖定

鎖定(Lock)

1. MySQL 使用下述方法將資料表上鎖
 - (1) 鎖定資料表：ISAM，MyISAM 及 Memory(Heap)資料表
 - (2) 鎖定資料頁：BDB 資料表
 - (3) 鎖定資料紀錄(row)：InnoDB 資料表
2. 鎖定(Lock)有兩種：
 - (1) Write Lock / Exclusive Lock (寫入/專屬鎖定)
 - ➔對象：不可讀/不可寫
 - ➔還有一種取得資源優先權較低的：Low Priority Write Lock
 - (2) Read Lock / Share Lock (讀取/共用鎖定)
 - ➔對象：可讀/不可寫◎資源取得優先權：W > R > LW
3. MySQL 在鎖定資料表時，可以完全排除互鎖(dead-lock)問題
 - (1) 對於寫入資料之上鎖方式，MySQL 是以下列演算法來處理：
 - ➔ 若該資料表未被鎖定，則立即上鎖。
 - ➔ 否則，將上鎖請求置入 write lock 佇列中。
 - (2) 對於讀取資料之上鎖方式，MySQL 是以下列演算法來處理：
 - ➔ 若該資料表未以寫入鎖定，則立即上鎖。
 - ➔ 否則，將上鎖請求置入 read lock 佇列中。

交易(Transaction)

「交易」是指將數個資料存取或更新的動作當成一個整體看待。

- ⇒ 出任何差錯，這些動作全部取消
- ⇒ 沒有差錯，這些交易的效果保證永久有效
- ⇒ All or Nothing

交易的特質→ACID

⇒ Atomicity (單元性)

將數個資料存取或更新的動作當成一個交易看待，而一個交易被視為一個不可分割的單元。

⇒ Consistency (一致性)

交易執行前和執行後，資料庫都應該是一致的。

⇒ Isolation (隔離性)

數個交易同時執行時，彼此互不干擾。

⇒ Durability (永久性)

一個交易內的任何一個資料存取或更新的動作沒有出現任何的差錯，則這些交易的效果保證永久存在。

交易並行處理(Concurrency control)

一般而言，DBMS 會同時交錯處理數個交易的運算動作。若無有效管控交易活動，則將可能發生以下狀況：

⇒ 更新遺失(lost update)

⇒ 讀取錯誤的資訊(dirty read)

⇒ 無法重複讀取(non-repeatable read)

⇒ 幽靈資料(phantom)

利用鎖(Lock)來產生正確的交易排程。

※實作 1

```
-- Clie A
use northwind;
set transaction isolation level read committed;
start transaction;

-- Clie B
use northwind;
set transaction isolation level read committed;
start transaction;
select productId, UnitsInStock from products where productId = 1;

-- Clie A
update products set unitsInStock = 11 where productId = 1;

-- Clie B
select productId, UnitsInStock from products where productId = 1 lock in share mode;

-- Clie A
commit;

-- Clie B
select productId, UnitsInStock from products where productId = 1;
commit;
```

※實作 2

```
-- Clie A
use northwind;
set transaction isolation level repeatable read;
start transaction;

-- Clie B
use northwind;
set transaction isolation level repeatable read;
start transaction;
select productID, UnitsInStock from products where productID = 1;

-- Clie A
update products set unitsInStock = 11 where productID = 1;

-- Clie B
select productID, UnitsInStock from products where productID = 1;

-- Clie A
commit;

-- Clie B
select productID, UnitsInStock from products where productID = 1;

-- Clie A
start transaction;
update products set unitsInStock = 12 where productID = 1;
commit;

-- Clie B
select productID, UnitsInStock from products where productID = 1;

-- Clie B
commit;
select productID, UnitsInStock from products where productID = 1;
```


MySQL 變數使用

※有三種變數

1. 一般變數：開頭以@標示。關閉連線後，即消失。
2. 系統、伺服器變數：開頭以@@標示。
3. 本地變數、預存程序的變數：宣告在 sp 裡面，並且只有在 sp 裡面有效。

※mysql4.1 之前，變數名稱大小寫有別；mysql5.0 開始就不區分變數名稱的大小寫。

※指定變數值

SET 用的指定運算子是=，但 SELECT 用的是:=。

```
mysql> SET @varname=3;  
mysql> SELECT @varname:=3;
```

※[實作]：查詢資料時，顯示資料行數

```
mysql> set @row_count = 0;  
mysql> select *, @row_count := @row_count + 1 as row_count from table_name;
```

MySQL 預儲程序的設計

※先天限制

Stored Routine 就是指 Stored Procedure 和 Function 。

以下就是 Stored Routine 及 Trigger 的限制：

1. SELECT 指令一定要用 INTO 子句將查詢結果回傳到變數中，否則出錯誤。另外，SHOW, EXPLAIN 和 CHECK TABLE 等指令不能使用。
2. 不能用 FLUSH 指令。
3. 不能遞迴使用 Stored Function 。
4. 不能修改已經被叫用的資料表，例如已經宣告 Cursor 的相關資料表，UPDATE 或 SELECT 指令引用的相關資料表，只要是在 Stored Routine 或 Trigger 有引用的資料表，都不能在 Stored Routine 或 Trigger 中有修訂的動作發生。
5. 在 Stored Routine 中建立的 Temporary Table，不能重覆宣告，不則會出現錯誤訊息: "Can't reopen table: 'tbl_name'" 。
6. Stored Function 會對引用的資料表做 Table Lock 的動作，但 Stored Procedures 不會。

以上大致說明了 MySQL 的 Stored Procedure 及 Function、Trigger 的一些限制。

※CURSOR 的使用

談到 Stored Procedure 和 Function 在資料庫中設計程式的時候，一定不能缺少 Cursors 的介紹，Cursor 是一個暫存在 Cache 中的資料集合，利用 Cursor 能將這個資料集合中的每筆資料錄進行固定的處理工作，這可以很方便的用在各種應用上，例如：可以利用 Cursor 來處理某個期間內已經結案的銷貨訂單，對每筆結案的銷貨訂單勾稽後，變更其狀態為結案，以防再拿來處理。

與 Cursor 相關的指令有四：

1. DECLARE: 宣告 Cursor 的資料結構及資料來源，使用 SELECT 指令來配合
2. OPEN: 把 Cursor 啟用並放到 Cache 中
3. FETCH: 由 Cursor 中讀取一筆資料錄
4. CLOSE: 關閉 Cursor，由 Cache 中移除 Cursor 的暫時資料集合及其定義