MySQL 基本操作环境

接下来几章介绍有关如何建立资料库、在资料库中查询与异动资料，会有在 MySQL 的指令操作模式，也有用 phpMyAdmin 的 Web 介面操作方式。

本章将只针对 MySQL 的基本环境使用和资料型态等做介绍。

开 离开

目录

11-1 MySQL 操作环境介绍

接下来几章介绍有关如何建立资料库、在资料库中查询与异动资料，会有在 MySQL 的指令操作模式，也有用 phpMyAdmin 的 Web 介面操作方式。

至於资料库的基本概念，譬如：关联式资料库的观念、何谓 SQL、正规化的步骤…等等，这些在第二章「关联式资料库与 SQL 概论」已介绍过。

本章将只针对 MySQL 的基本环境使用和资料型态等做介绍。

11-1 MySQL 操作环境介绍

从 Windows 左下角的「开始」功能表进入，执行「 AppServ 」下的「 MySQL Command Line Client 」来进入 MySQL。

若在安装时有设定 MySQL 管理者（ root ）的密码，那么要输入正确的管理者密码才能成功进入 MySQL ：
linux 下命令行输入： mysql -u root -p 进入；和 windows 操做都基本相同；

11-1 MySQL 操作环境介绍

当成功进入 MySQL 之后，它的命令列会变成以 “ mysql>” 为开头。每下完一个 MySQL 指令后要加分号「；」再按 Enter 键才会执行该指令。

如果要重覆执行之前所下过的指令，或想把之前执行过的指令叫出来修改再执行，可以按上下键来浏览前后一个已下过的指令。若要离开 MySQL 的环境，只要下 “ exit; ” 指令即可。

11-1 MySQL 操作环境介绍

几个在 MySQL 比较有机会用到的指令：

- 显示目前已有的资料库 - show databases

show databases 指令可以显示目前伺服器上已有的资料库名称。

11-1 MySQL 操作环境介绍

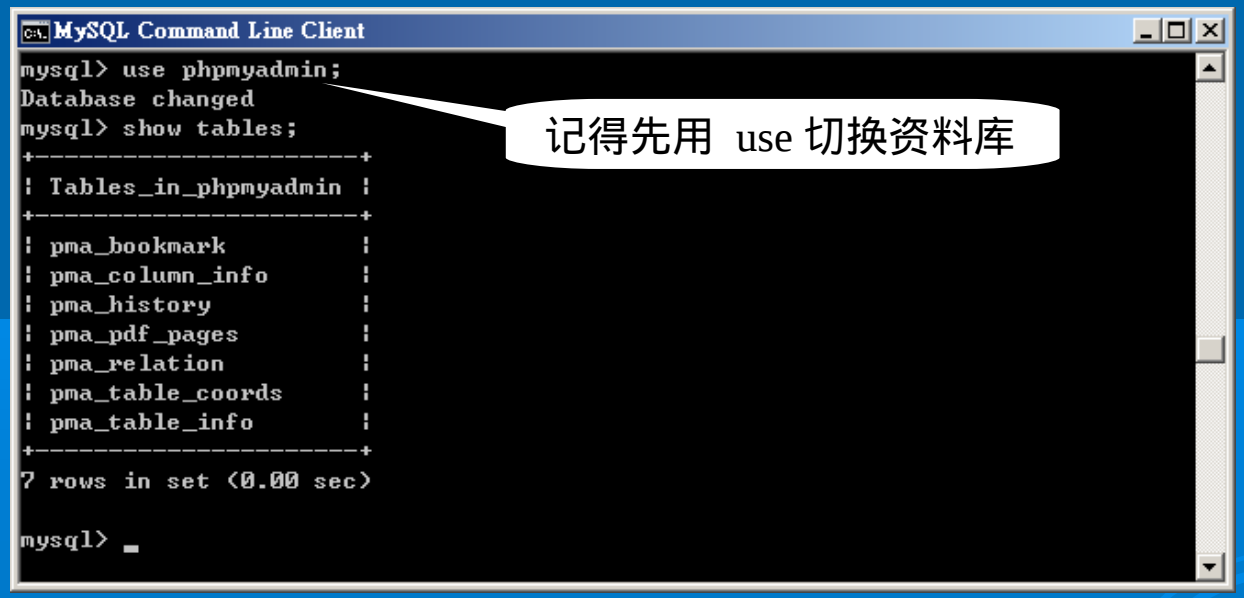
- 建立资料库 - create database [databasename]

create database 指令可以建立新的资料库，后面加建立的新资料库名称即可。

11-1 MySQL 操作环境介绍

- 显示资料库已有的表格 - show tables

在用 use 指令切换资料库之后，可以用 show tables 来查询目前所在的资料库中有哪些表格。

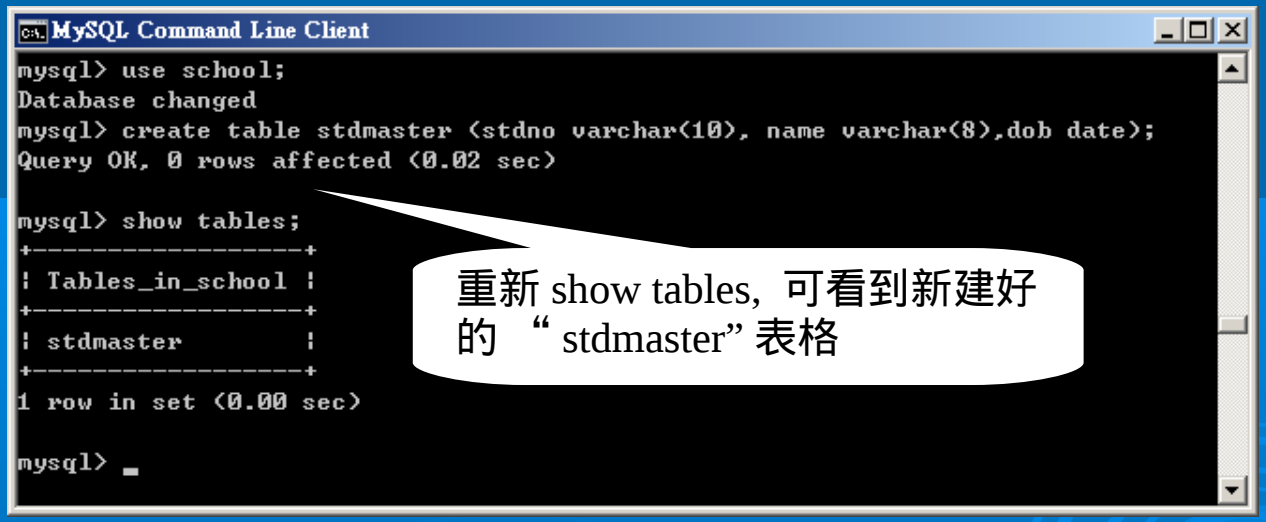


11-1 MySQL 操作环境介绍

- 建立表格 - create table

建立表格的 create table 指令格式如下：

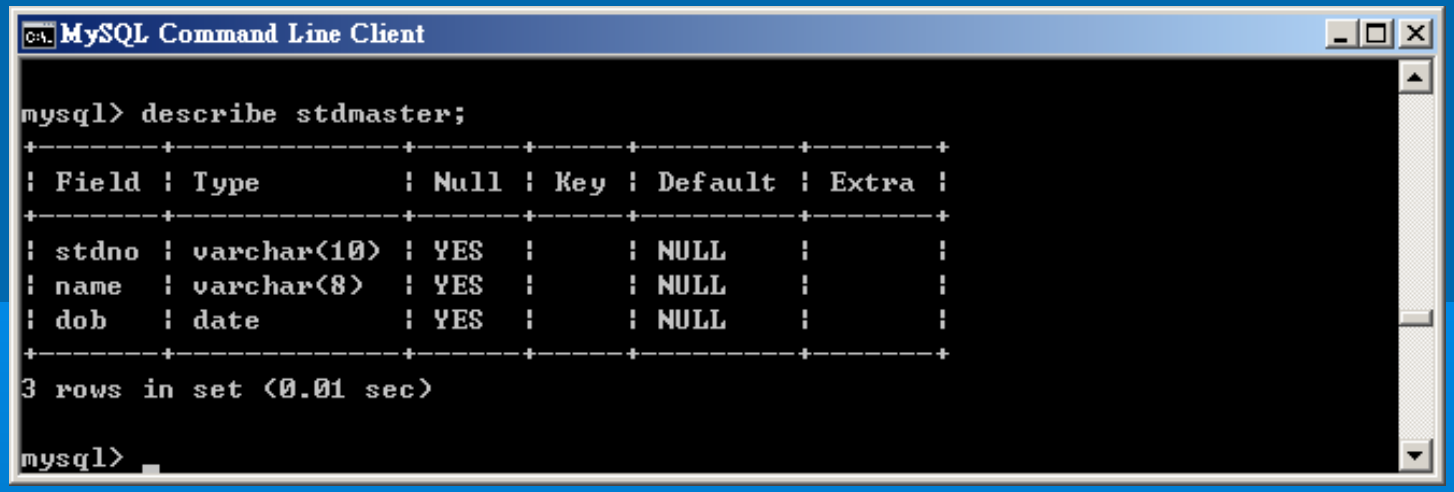
create table [table name]
(field1 type1, field2 type2, …..);



11-1 MySQL 操作环境介绍

- 显示表格的栏位定义 - describe [table name]

describe 指令后加上表格名称，可以显示该表格的栏位名称和资料型态…等有关表格结构（ Structure ）的定义。



11-1 MySQL 操作环境介绍

- 删除表格 - drop table [table name]

drop table 指令可以将指定的表格，连同表格内的所有资料记录一并删除。在执行时也不会有警告讯息，要小心使用。

11-1 MySQL 操作环境介绍

- 删除资料库 - drop database [database name];

drop database 指令会将目前存在的资料库整个删除，后面加欲删除的资料库名称即可。

要特别注意的是，要是资料库里已经有些表格了， drop database 就会自动也将资料库里所有的表格，连同表格内的资料一并删除。在执行时不会有警告讯息，使用上要小心。

- 切换（ Switch ）目前使用的资料库 - use [db name]

所谓切换目前使用的资料库，就是要将某个资料库视为目前所有指令（如建立、删除表格与显示资料库已有的表格）所针对的资料库对象。

譬如：建立表格之前，要先切换作用中的资料库，这样稍后所建立的表格才会建立在想要放的资料库里。

11-2 建立表格

建立表格的指令是 Create Table，这一节将介绍以 Create Table 指令建立新表格的语法。

11-2-1 建立表格 - Create Table

完整的 Create Table 指令语法有许多参数可选用，这里仅列出最常用的部份供读者参考。

Create Table <table name>
[<field_definition,...>]
[select_statement];

其中 field_definition:

fieldname type
[NOT NULL | NULL]
[DEFAULT default_value]
[AUTO_INCREMENT [PRIMARY KEY]]
[ZEROFILL]

11-2-1 建立表格 - Create Table

说明： field_definition 即栏位定义，它包含了：栏位名称、栏位型态、和它的栏位设定子。 MySQL 有许多栏位设定子可使用，在下一小节再介绍。

在建立表格时搭配 select_statement 是代表在建立表格的同时，将别的表格的资料直接复制过来。

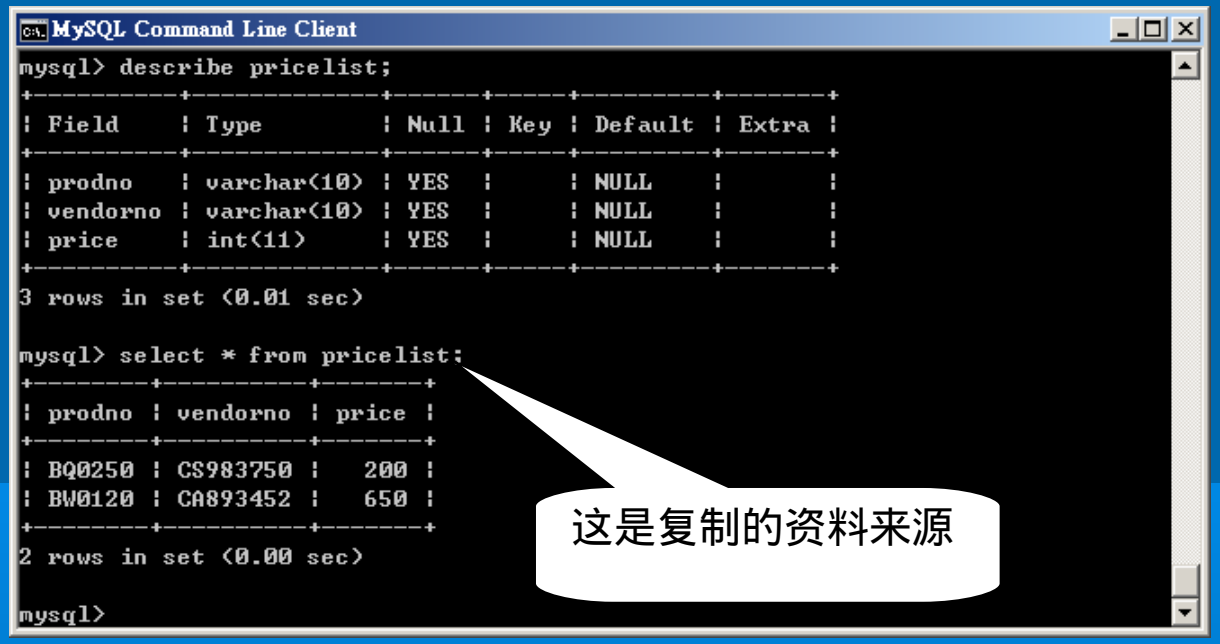
若使用 select_statement 可以省略 field_definition ，建立出来的表格结构，即栏位定义，就会和资料来源的表格一样。使用 select_statement 仍然可以自行定义新表格的结构，注意若栏位长度较来源资料短，资料会有被截断问题。

范例：

```
create table pricelist_new (prodno varchar(5), vendorno varchar(5),
        price smallint) select * from pricelist;
```

11-2-1 建立表格 - Create Table

以下是复制资料来源的表格 pricelist 目前的表格结构和内容：

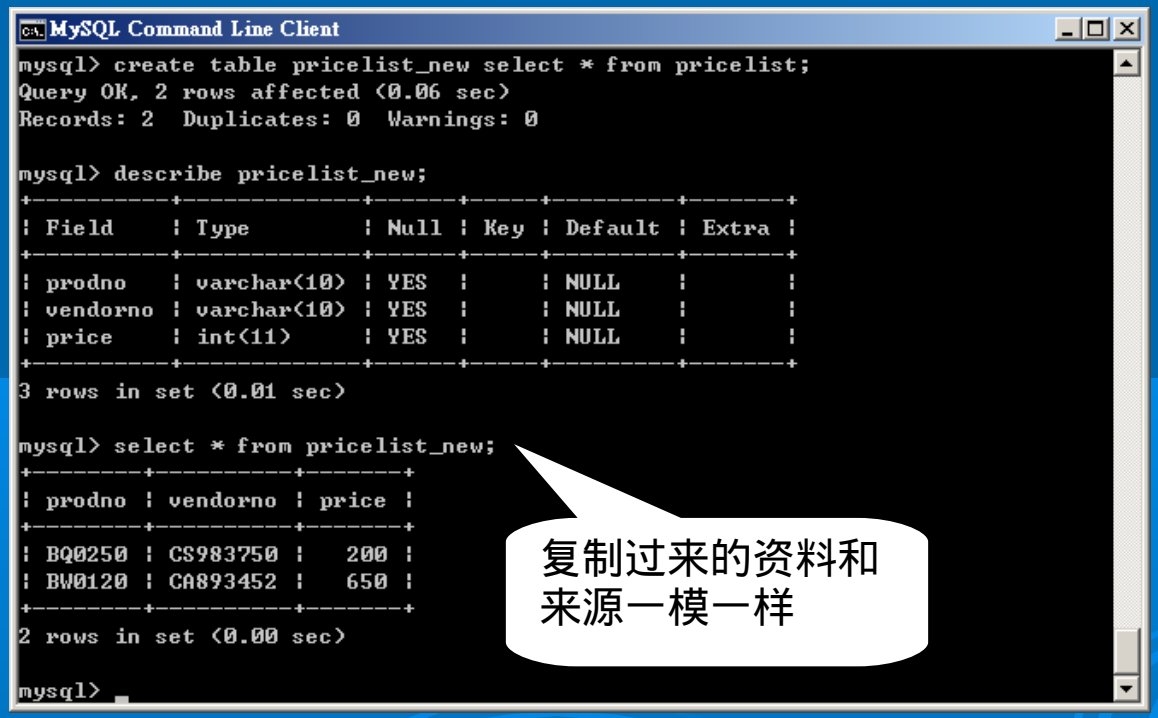


11-2-1 建立表格 - Create Table

经过执行指令建立新的表格 pricelist_new，并将表格 pricelist 的资料复制给它之后，新的表格 pricelist_new 结构因为文字的长度只有 varchar(5)，而非 pricelist 表格中定义的 varchar(10)，所以文字的后半段被截掉了。

11-2-1 建立表格 - Create Table

但若在 Create Table 时，不指定栏位定义，就没有上述问题。因为建立的新表格结构会和来源表格一模一样，资料也一样，当然资料也不会有被截掉的问题。使用方式视情况而定。



11-2-2 栏位设定子

在定义表格的栏位时，除了一定要定义每个栏位的名称和资料型态，My SQL 还提供了几个资料栏位设定子（ Column Modifier ）。

它可以用来设定资料栏位的属性，包括了预设值（ Default ）、自动增量（ Auto Increment ）、无号数（ Unsigned ）、主键值（ Primary Key ）等。

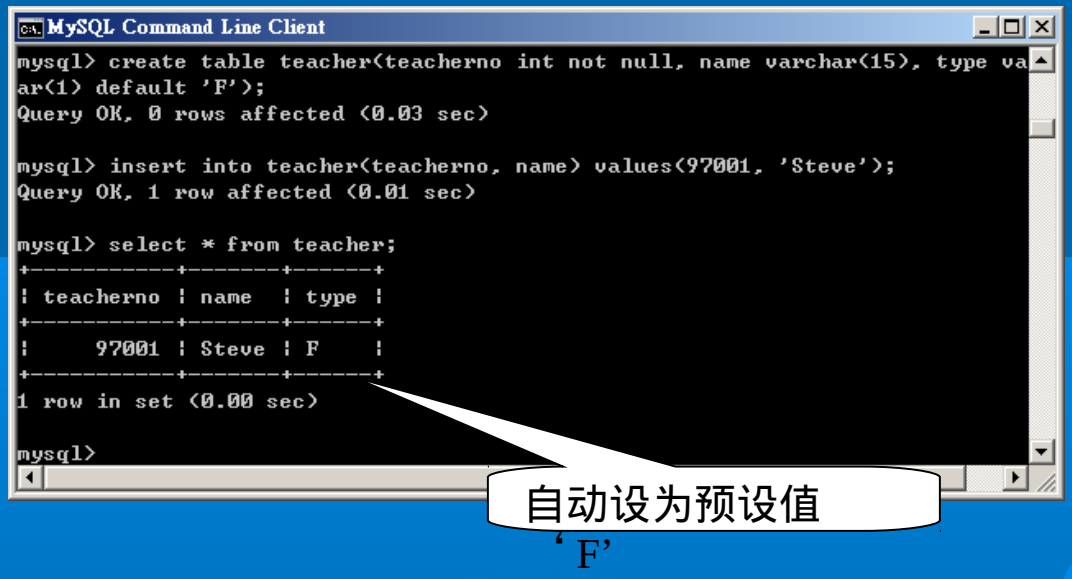
- 空值 / 非空值（ Null/Not Null ）

Null 是允许该栏位的值是空值，而 Not Null 是指该栏位不允许接受空值。没有特别设定的话，预设状态是 Null 。

11-2-2 栏位设定子

- 栏位预设值（ DEFAULT ）

若某个栏位有用“ DEFAULT” 设预设值，那么在新增（ Insert ）记录时，若没有针对该栏位设定要给予的值，那么该栏位就会自动以预设值做为此次新增记录时的栏位值。



11-2-2 栏位设定子

- 自动增量设定（ AUTO_INCREMENT ）

“AUTO_INCREMENT” 自动增量设定子只可用在整数型态栏位。栏位如果加了自动增量设定，则每增加一笔新纪录，不需要为该栏位设定值，它的值就会自动增加 1 。

当你插入一笔资料时，其中的设定为“ AUTO_INCREMENT” 栏位值必须存入 NULL 值、 0 或空白值。也可指定存入一个数值，但是假如该数字已经存在的话，则会产生错误。

假如这次所要插入的记录，其栏位值将成为目前资料表格中该栏位的最大值，则再一次增加的纪录，其该栏位值将会以此次加入的值加 1 。

11-2-2 栏位设定子

当新建一个新的资料表，并且指定其中一个栏位为“AUTO_INCREMENT”时，可以在插入第一笔资料时即指定一个值做为“AUTO_INCREMENT”的初始值，之后每次新增记录就可以自初始值开始递增。当一个栏位被设为“AUTO_INCREMENT”，且它是唯一个“自动增量栏位，MySQL 会要求将它设为 Primary Key。

例如：

```
Create Table custmast
  (custno int not null auto_increment primary key,
birthdate date);
```

11-2-2 栏位设定子

- 自动补齐零（ ZEROFILL ）

“ZEROFILL” 适用於设在所有数值型态的栏位。如果将一个数值型态栏位设定为 “ ZEROFILL”，则会在数值之前补齐 “ 0”。例如，在插入记录到资料表格时，将 78 的值存入到一个宣告为 INT “ZEROFILL” 的栏位时，此时存放的资料为 0000000078 。
“ZEROFILL” 这个设定子对於需要格式化输出的数值资料有帮助，譬如，会员编号从 0000000001 、 0000000002 、 0000000003 、 … . 开始编号下去。前面不足位数的部份会用零补齐，这样相同的位数会自动对齐好，在浏览资料时也较清楚。

例如：

```
Create Table custmast
  (custno int zerofill, address varchar(30));
```

11-2-2 栏位设定子

- 无号数（ UNSIGNED ）

无号数（ UNSIGNED ）这个栏位设定仅适用於整数型态栏位。 一般的数字都有正负号，所以在二进位表示下，最高的位元就被拿来当作符号位元，所以才允许有负数。所谓无号数就是舍弃符号位元，改拿来表现数字，因此它只表示正数。因为多了一个位元可使用，它可表示的正数范围是有号数的两倍。这在某些数值只需要正数而不需要负数的情形下可以更加有效利用储存空间。

```
Create table sales
      (prodno varchar(10),
   pirce int unsigned,
amount int unsigned);
```

11-3 MySQL 资料型态

资料库的资料型别（ Data Type ）是指一个资料栏位内所存放资料之型态，它关系著资料所占磁碟空间的大小与查询资料库时的执行效率。

而这节要介绍 MySQL 资料库栏位可储存的资料型态，这样才能知道如何在建立资料库表格时，为表格的每个栏位选择适合它的资料型态。MySQL 资料库的栏位资料型态，大致可分为五种：

- 数字型态
- 日期与时间型态
- 字串与字元型态
- 列举型态
- 集合型态

11-3 MySQL 资料型态

将这几个资料型态的储存空间需求及其值的大小范围整理如下：

■数字型态

各种数值型态所用之记忆体空间、数值范围如下所示， M 表示数字的位数， D 表示小数点以下的位数：

型態	儲存空間需求	範圍
TinyInt [(M)]	1 byte	-2^{7}~2^{7}-1
SmallInt [(M)]	2 bytes	-2^{15}~2^{15}-1
MediumInt [(M)]	3 bytes	-2^{23}~2^{23}-1
Int, Integer [(M)]	4 bytes	-2^{31}~2^{31}-1
BigInt [(M)]	8 bytes	-2^{63}~2^{63}-1
Float [(M)]	4 bytes	-1.1E+38~3.4E+38 (約)
Float	4bytes	-1.1E+38~3.4E+38 (約)
Double[(M,D)], Real [(M,D)]	8 bytes	-1.79E+308~1.79E+308 (約)
Decimal [(M,D)] Numeric [(M,D)]	依據 M 與 D 值而定，為可變長度	

11-3 MySQL 资料型态

- 日期与时间型态

日期与时间资料型态，包含下列型态，其中 y 为年份， m 为月份，d 为日期， h 表小时， m 表分钟， s 为秒：

型態	意義	儲存空間需求	零值
Date	日期 (yyyy-mm-dd)	3 bytes	'0000-00-00'
Time	時間 (hh:mm:ss)	3 bytes	'00:00:00'
DateTime	日期與時間組合 (yyyy-mm-dd hh:mm:ss)	8 bytes	'0000-00-00 00:00:00'
TimeStamp	yyyymmddhhmmss	4 bytes	'00000000000000'
Year	年份 (yyyy)	1 byte	0000

11-3 MySQL 资料型态

- 字串与字元型态

字串与字元型态，包含下列资料型态，其中 N 表字串的长度：

型態	儲存空間需求	最大長度
Char(N)	N bytes	1 <= N <= 255
VarChar(N)	N+1 bytes	1 <= N <= 255
TinyBlob, TinyText	N+1 bytes	L < 28
Blob, Text	N+2 bytes	L < 28
MediumBlob, MediumText	N+3 bytes	L < 224
LongBlob, LongText	N+4 bytes	L < 224

11-3 MySQL 资料型态

- 列举型态

型態	儲存空間需求	最大長度
Enum	1 或 2 bytes	65535 個成員

- 集合型态

型態	儲存空間需求	最大長度
Set	1、2、4、8 bytes	64 個成員

11-4 资料处理相关的 SQL 指令介绍

第二章已介绍过 SQL 语法的发展历史，以及 SQL 指令可以分成 DDL（资料定义语言）、 DML（资料处理语言）、 DCL（资料控制语言）三大部份。

这一节主要介绍其中 DML 中的资料查询指令 - Select 以及有关资料异动的指令 - Insert/Delete/Update 。对於不曾接触过 SQL 指令的读者，

这一节的内容对您来说是非常的重要，务必充分了解之后再往下阅读。基本的 SQL 指令其实相当简单易学，接下来的介绍都会以例子说明，读者可以很快理解。

11-4-1 资料查询 – Select

Select 是所有 SQL 指令中最常使用到的指令，它是向资料库的一个或多个表格所下的资料查询（ Query ）指令。透过 Select 指令可以筛选出所需要的部份资料记录（ Records ），还可以指定查询资料结果显示的次序。也可利用表格之间的关联将不同表格的资料串连（ Join ）在一起。

虽然在不同资料库产品中， SQL 指令语法会有些差别，但在本书介绍的基础指令上并不会有什么差别。在 Access, SQL Server, Oracle, 甚至在下一章要介绍的 phpMyAdmin ，都提供可以让使用者直接下 SQL 指令的地方，基本的语法并无差异，对各种资料库都可适用。

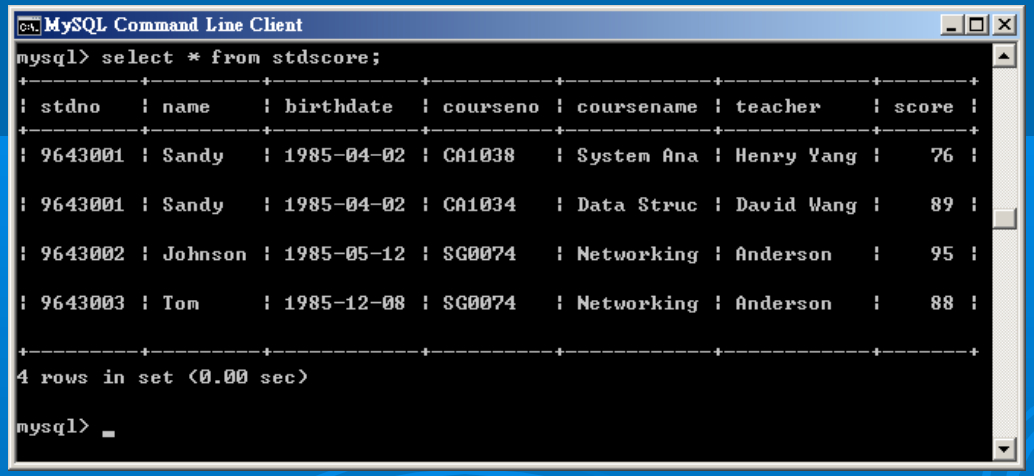
接下来所用的范例 SQL 指令，读者也可以在其他资料库软体上使用，只要资料一样，显示出来的结果也会一样，只是不同软体的画面不相同而已。这一节将以 MySQL 的操作画面作说明。

11-4-1 资料查询 – Select

- Select * from [table name]

说明： Select * 代表要将表格内所有的栏位都显示出来。 “ *” 代表所有的栏位名称。

范例： select * from stdscore;
它会将 “ stdscore” 表格的所有栏位全部显示出来。



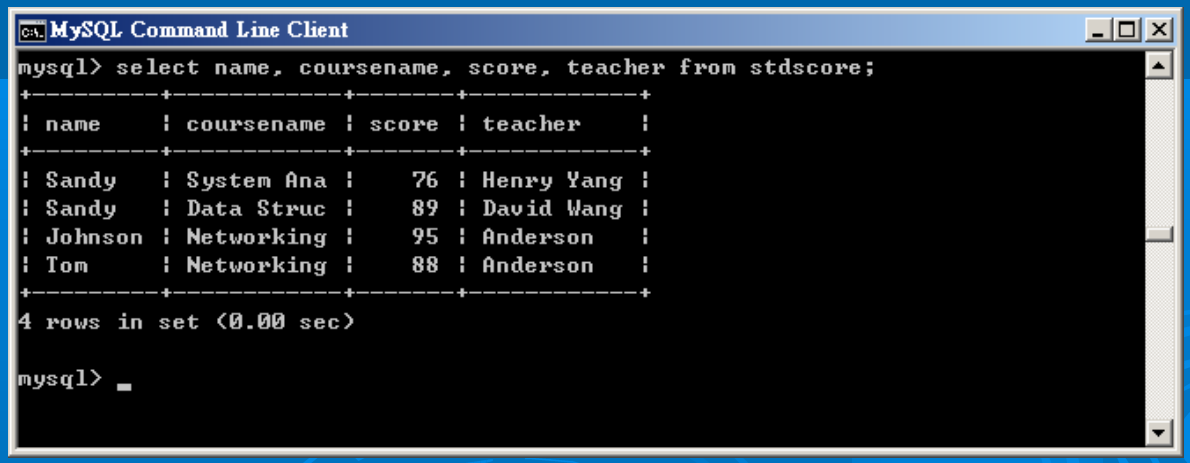
11-4-1 资料查询 – Select

- Select field1[, field2, …] from <table name>;

说明：只将 Select 后所列的栏位按照指令里的次序显示出来。

范例： select name, coursename, score, teacher from stdscore;

它会将“ stdscore” 表格内的
name、 coursename、 score、 teacher 栏位次序显示出来。



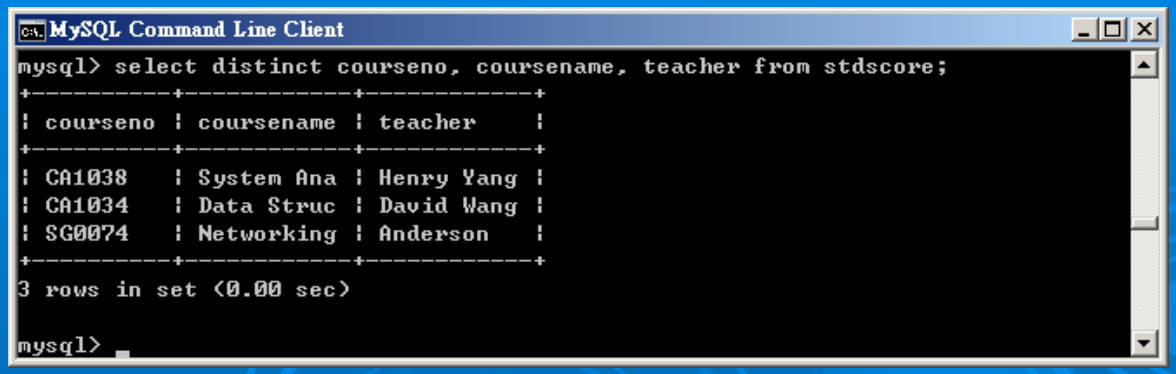
11-4-1 资料查询 – Select

- Select distinct field1[, field2, …] from <table name>;

说明： distinct（不同的）的作用是在将栏位所有相同值各取唯一的一笔出来。若 distinct 是摆在很多栏位之前，意思是将表格中 Select distinct 后所列的几个栏位所有不同值的排序组合都显示出来。

范例： select distinct courseno, coursename, teacher from stdscore;

它会将“stdscore”表格内所有资料 name, coursename, score, teacher 栏位值的不同组合都显示出来。



11-4-1 资料查询 – Select

- Select field1[, field2, …] from <table name>;
  where <query_condition>

说明： query_condition 指定了要筛选满足哪些逻辑条件成立的记录（ Records ）。常见的包括等於（ = ）、不等於（ <> ）、大於（ > ）、大於或等於（ >= ）、小於（ < ）、小於或等於（ <= ）…等的比较条件。

也可以用“ IN” 来指定某个栏位的值是落在哪些范围以内才要筛选出来。或用“ BETWEEN…AND…” 来指定某栏位的值是要落在所指定的最小与最大值范围区间内。“ LIKE” 和万用字元符号“ %” 搭配使用则是用来做文字型态资料的筛选。

最后，可以用“ AND” 和“ OR” 条件将多个逻辑条件一起写在同一个查询条件里，达到多重筛选的目的。而“ AND” 的运算次序高於“ OR” 。

11-4-1 资料查询 – Select

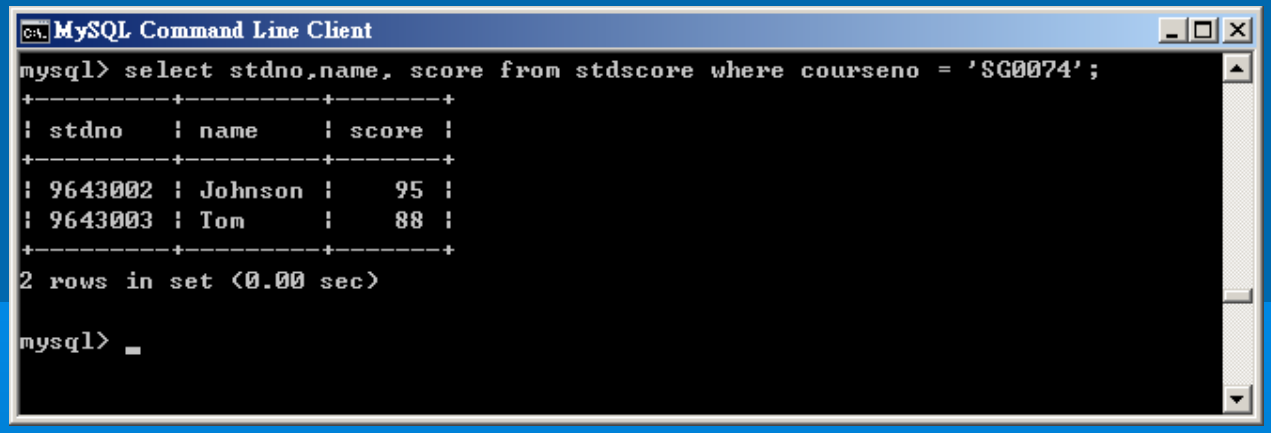
若 Select 后不是用“ *” 符号，而是只列出所要的部份栏位出来。其实就和第 2-5-2 小节「关联运算」中所介绍的投影运算（ Project ）的观念很类似，作用是在表格中只筛选出所需要的栏位（ Field ）资料出来。

至於 Select 加上了 where 条件，则是和第 2-5-2 小节「关联运算」中所介绍的选取运算（ Select ）的观念很类似，作用是在表格中只筛选出符合所要的资料记录（ Record ）出来。

11-4-1 资料查询 – Select

范例： select stdno, name, score from stdscore where courseno=’SG074’ ;

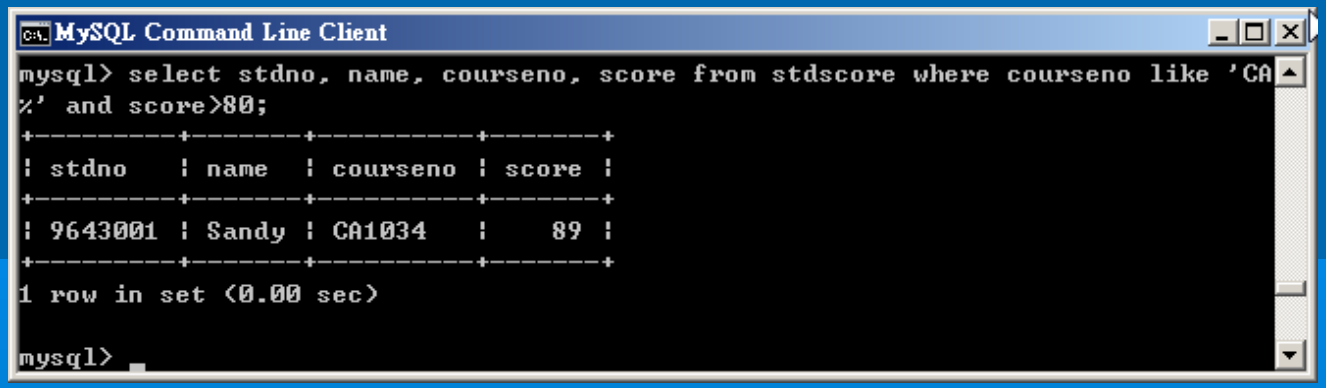
它会将课程代号是’ SG074’的学生学号、姓名和分数显示出来。



11-4-1 资料查询 – Select

范例： select stdno, name, courseno, score from stdscore where courseno like ’CA%’ and score>80 ;

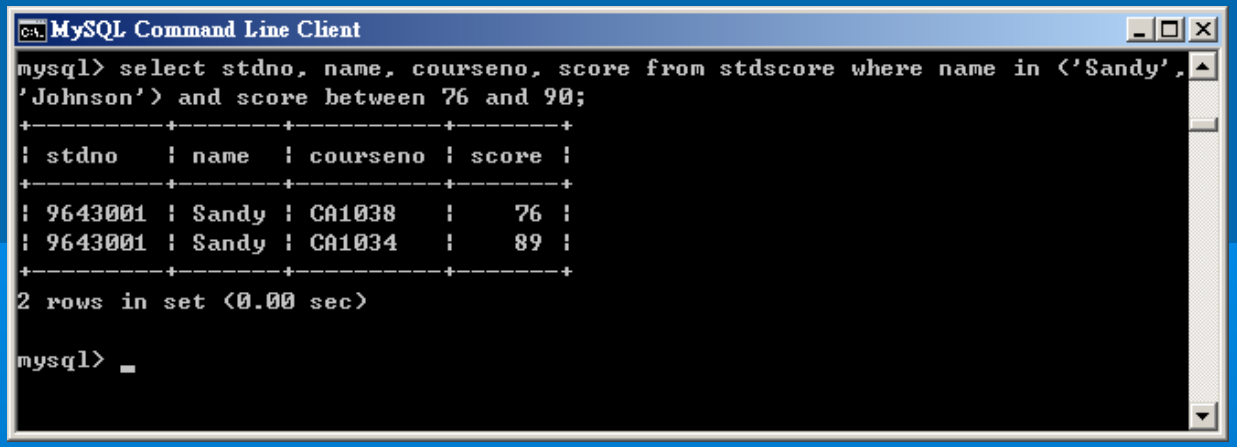
它会将课程代号是 ’ CA’ 开头，而且分数超过 80 分的学生学号、姓名、课程代号和分数显示出来。



11-4-1 资料查询 – Select

范例： select stdno, name, courseno, score from stdscore where name in ('Sandy','Johnson') and score between 76 and 90;

它会将学生姓名是“Sandy”或“Johnson”，而且分数介於 76（包含）到 90（包含）之间的学生成绩资料显示出来。



11-4-1 资料查询 – Select

- Select field1[, field2, …] from <table name>;
order by sort_field1 [desc] [,sort_field2 [desc], …]

说明： order by 是指将查询出来的资料记录依照 order by 后的排序栏位由小至大排序。若先后列出多个排序栏位，表示先依照第一个排序栏位由小至大排序，若第一个排序栏位的值相同，则再视第二个排序栏位的值由小至大排序，依此类推。

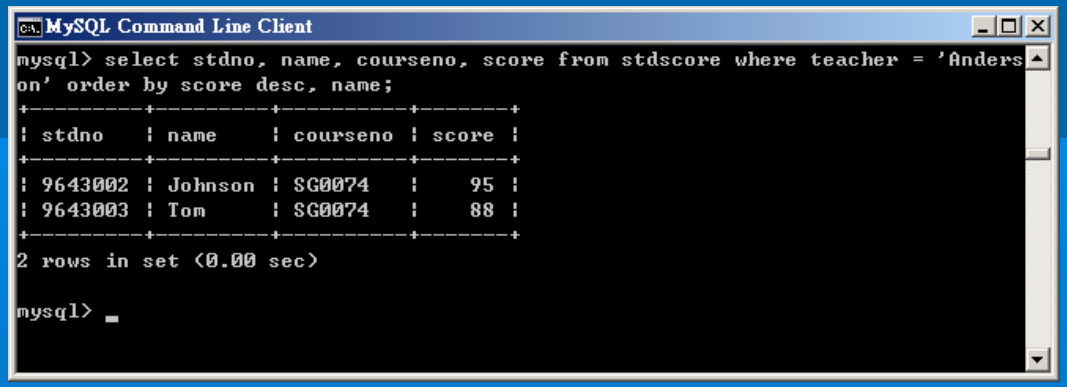
若在排序栏位名称后加上“ desc”，则表示该栏位的排序是由大至小（ Descending ）。每个排序栏位的排序方式可以不相同，可视需要个别指定每个排序栏位的排序方式，若没特别指定则会用预设方式：由小至大排序。

order by 可以和 where 条件一起使用，代表要先用 where 筛选出符合条件的资料记录，再依 order by 的次序显示筛选后的结果。但要注意在 SQL 指令中要先写 where 子句再用 order by 子句。

11-4-1 资料查询 – Select

范例： select stdno, name, courseno, score from stdscore
where teacher = 'Anderson'
order by score desc, name;

它会将授课讲师是“Anderson”的学生成绩资料显示出来，显示时要先依照分数由高至低排序，若分数相同再依照学生姓名由小至大排序。



11-4-1 资料查询 – Select

- Select [group_field1 [,group_field2, …]], count （ * ）
  from <table name>
  group by group_field1 [,group_field2, …]]
  [having having_condition];

说明： group by 的作用是将整个表格资料中，被列为汇总栏位的一个或多个栏位的所有值的排列组合情况加以分组（ Grouping ）。 count 是 SQL 可用的函数之一， count(*) 是该分组的资料笔数。

若 Select 指令有用到 group by 子句，那么写在 Select 后面的栏位应是这些分组栏位之中的部份栏位。

Select 后也可用 count 或接下来将介绍的 avg 或 sum 等汇总函数的栏位。

11-4-1 资料查询 – Select

至於 having group_condition 则是限定只有分组出来的结果有满足分组条件的，才显示该分组的资料。

having_condition 中也必须使用汇总函数的栏位，譬如：count(*)>1 ，或接下来介绍的其它汇总函数来表示，

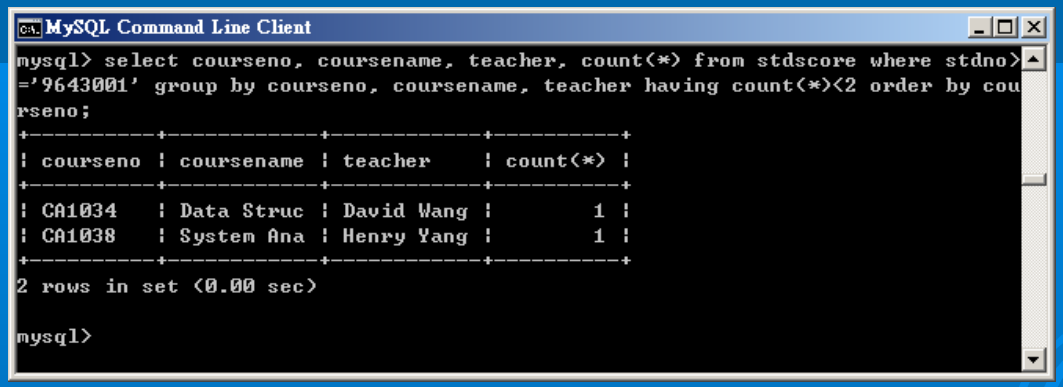
譬如： avg(score)>80 。若省略 having 子句，则所有分组结果都会显示出来。

group by ... having … 可以和 order by 和 where 子句一起并用，但在 Select 指令中的次序是先 where ，接下来是 group by ... having … ，最后才是 order by 。

11-4-1 资料查询 – Select

范例： select courseno, coursename, teacher, count(*) from stdscore where stdno>='9643001' group by courseno, coursename, teacher having count(*)<2 order by courseno;

它会将学生代号大於或等於“ 9643001” 的学生成绩中，课程代号、课程名称、授课讲师的所有不同排序组合显示出来，也用 count(*) 来显示分组笔数。但最后只将其中分组笔数小於 2 笔的分组资料才显示出来。



11-4-1 资料查询 – Select

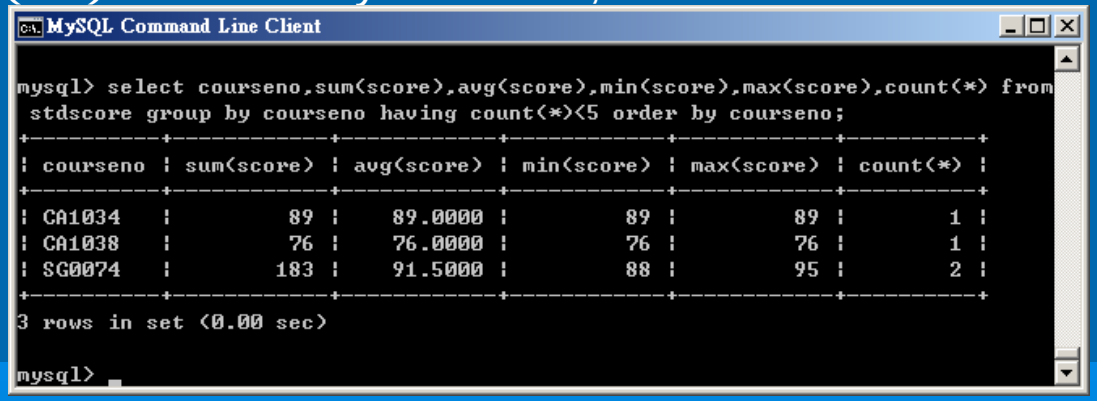
- Select sum(field)/avg(field)/min(field)/max(field) from <table name>
  group by group_field1 [,group_field2, …]
  [having having_condition];

说明： group by 除了可搭配前面所提的 count 函数，也可使用 sum、 avg、 min、 max 这些汇总函数，用来取得各分组的加总、平均、最小值、最大值。

having_condition 也必须是用汇总函数的栏位，譬如： count（ * ）>1、avg（ score ）>80 或 min（ score ）>80… 等。若省略 having 子句，则所有分组结果都会显示出来。

11-4-1 资料查询 – Select

范例： select courseno,sum（score）,avg（score）,min（score）,max（score）,count（*） from stdscore group by courseno having count（*）<5 order by courseno;



11-4-1 资料查询 – Select

- Select table1.field1 [, table1.field2…], table2.field1 [,table2.field2…],
  from <table1>, <table2> where <table1>.foreignkey = <table2>.keyfield;

说明：这是两个表格的合并，透过第一个表格的外来键和第二个表格的主键来做连结。

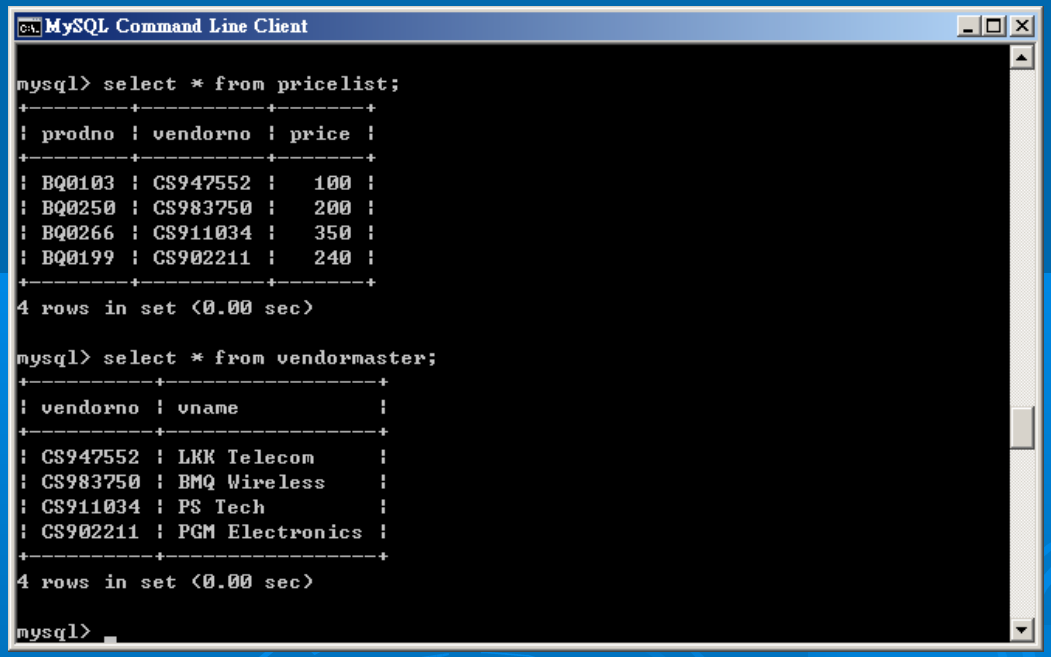
若表格名称太长，可以为表格名称设别名（ Alias ），语法如下：

Select alias1.field1 [,alias1.field2…], alias2.field1 [,alias2.field2…],
from table1 alias1, table2 alias2
 where alias1.foreignkey = alias2.keyfield;

11-4-1 资料查询 – Select

- 范例： select p.*, v.vname from pricelist p, vendormaster v
  where p.vendorno=v.vendorno;

假设有产品价格主档表格“ pricelist” 和供应商主档资料表格“ vendormaster” 的资料内容如下：



11-4-1 资料查询 – Select

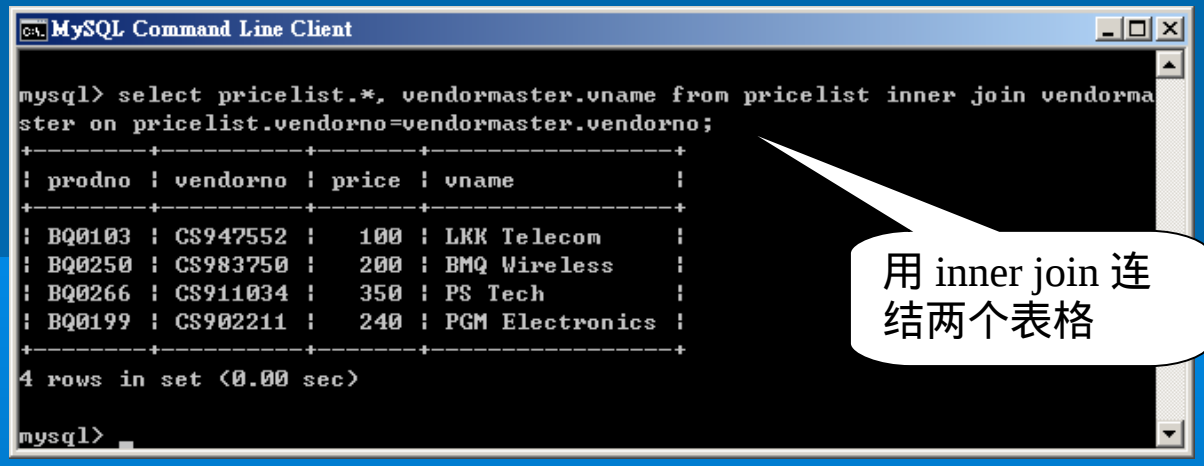
将这两个表格资料连结后，结果如下：

另外，也可以用 inner join 和上面的指令达成相同的表格连结效果。

11-4-1 资料查询 – Select

范例： select pricelist.*, vendormaster.vname from pricelist inner join vendormaster on pricelist.vendorno=vendormaster.vendorno;

执行的结果和前面用 where 条件的方式一样：



11-4-2 资料异动 – Insert、 Delete、 Update

表格的资料内容异动主要有三种方式：

- 插入（ Insert ）：在表格中加入一笔新的资料记录。
- 删除（ Delete ）：在表格中移除已存在的资料记录。
- 更改（ Update ）：更改在表格中已存在的资料记录的栏位值。

11-4-2 资料异动 – Insert、 Delete、 Update

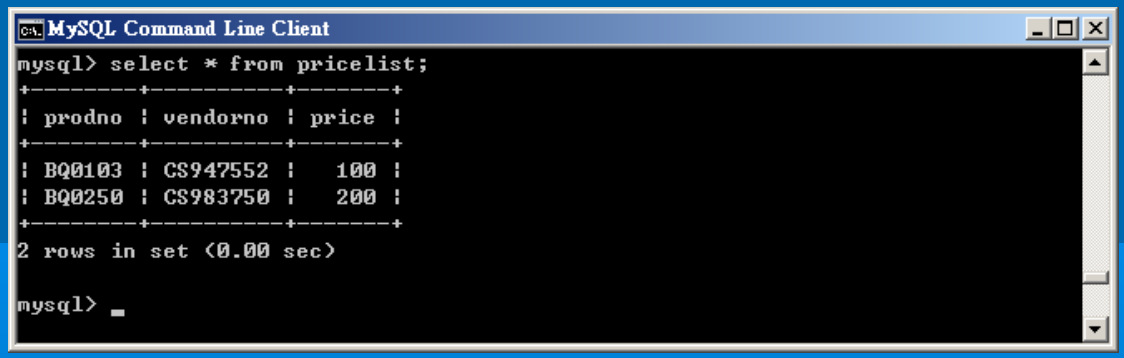
- Insert Into <table nane> [(field1[, field2, …)]]
  Values (value1[,value2, …)] ）;

说明：在表格中插入一笔新的记录，并将指定的栏位值依指定的栏位名称次序填入对应的栏位中。若在表格名称后省略了栏位名称次序，那么在 values 后的小括号内就要依照表格定义的栏位次序一一填入所有的栏位值。若填的栏位值的次序有误，很可能会造成资料错置或资料型态不符合的错误。

11-4-2 资料异动 – Insert、 Delete、 Update

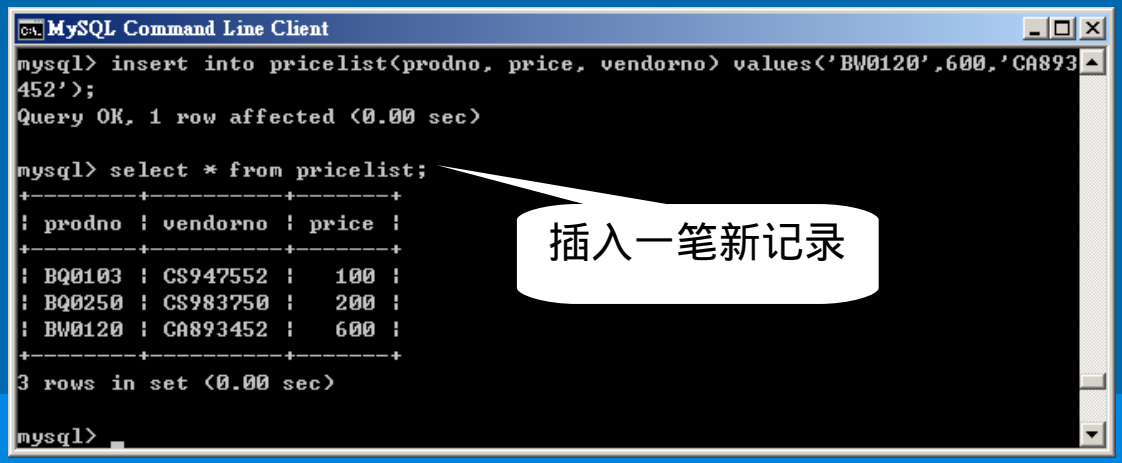
范例： insert into pricelist(prodno, price, vendorno)
values('BW0120',600,'CA893452');

假设产品价格表格“ pricelist” 目前有两笔记录，资料内容如下：



11-4-2 资料异动 – Insert、 Delete、 Update

在执行完 insert 指令后变成三笔记录，资料内容如下：



11-4-2 资料异动 – Insert、 Delete、 Update

- Delete from [table nane] [where delete_condition] ;

说明： Delete 指令会删掉所指定的符合删除条件的记录，其中 delete_condition 是删除条件。若省略了删除条件，则会将整个表格的资料统统删除，使用上要留意。

范例： delete from pricelist where vendorno='CS947552';
接续前一个例子的结果，再删除供应商代号是 “ CS947552” 的记录之后，结果只剩两笔记录，内容如下：

11-4-2 资料异动 – Insert、 Delete、 Update

- Update <table nane>
  Set field1=values1 [, field2=values2….]
  [where update_condition] ;

说明： Update 指令会更改掉所指定符合更新条件的记录，其中 update_condition 是更新条件，而且只有列出来要更新内容的栏位才会做更改，其余栏位内容不变。若省略了更新条件，则会将整个表格的统统更新，使用上也要留意。

11-4-2 资料异动 – Insert、 Delete、 Update

范例： update pricelist set price =650
where vendorno='CA893452' and price = 600;

接续前一个例子的结果，对於供应商代号是 “CA893452” 且价格是 600 的记录，将其价格变更为 650。变更后内容如下：

SQL 語法 -CREATE

創造表格或是資料庫

- CREATE DATABASE db_name
- CREATE TABLE tbl_name (definition)

- CREATE DATABASE lalala
- CREATE TABLE lalala(name text,age int,birthday data)

SQL 語法 -UPDATE

更新資料表內容

- UPDATE tbl_name SET col1=XXX,col2=XXX… (WHERE XXX)

- UPDATE tbl_name SET age=19
- UPDATE tbl_name SET age=19 WHERE name=‘abaw’

SQL 語法 -SELECT

選取欄位

- SELECT col1,col2… FROM tbl_name (WHERE XXX) (ORDER BY XXX) (LIMIT X,X)

- SELECT * FROM tbl_name
- SELECT * FROM tbl_name WHERE name=‘abaw’
- SELECT name,age FROM tbl_name ORDER BY age
- SELECT * FROM tbl_name LIMIT 5,10
- SELECT * FROM tbl_name LIMIT 1

SQL 語法 -INSERT

插入一筆新資料

- INSERT INTO tbl_name (col1,col2,…) VALUES(data1,data2,…)
- INSERT INTO tbl (name,age) VALUES(‘abaw’,19)

SQL 語法 -CREATE

創造表格或是資料庫

- CREATE DATABASE db_name
- CREATE TABLE tbl_name (definition)

- CREATE DATABASE lalala
- CREATE TABLE lalala(name text,age int,birthday data)