資訊管理學系 陳士杰老師

資料庫系統管理
Database System Management

SQL：結構化查詢語言
SQL: Structured Query Language

國立聯合大學
NATIONAL UNITED UNIVERSITY

Outlines

- Database Languages
- SQL資料型態
- SQL指令的種類：
  - Data Query Language, DQL (資料查詢語言)
  - Data Definition Language, DDL (資料定義語言)
  - Data Manipulation Language, DML (資料處理語言)
  - Data Control Language, DCL (資料控制語言)
  - 其它

【講義：Ch. 6】

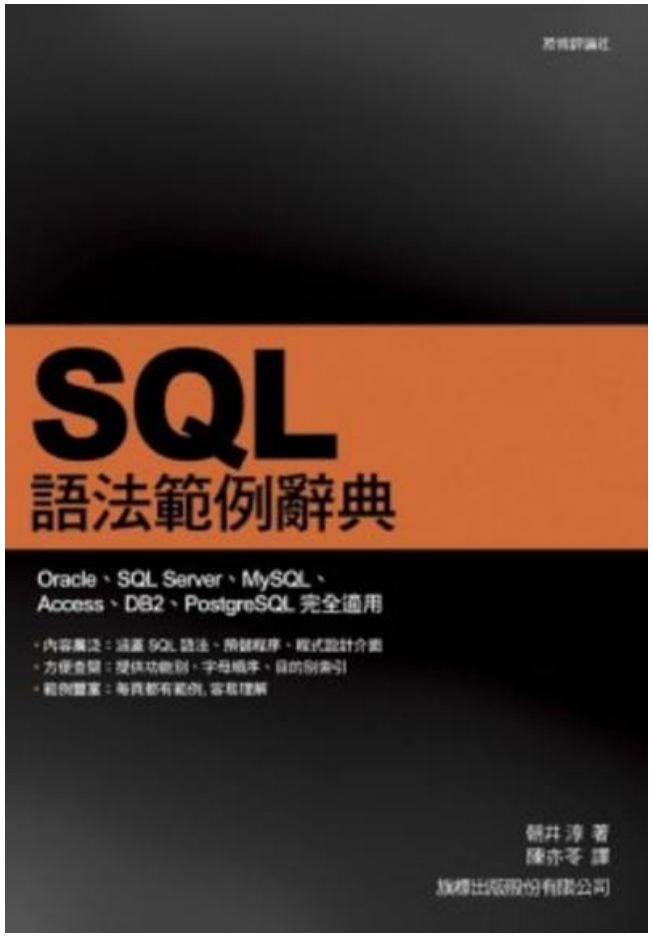
【原文：Ch. 8, Ch. 9】

Database Languages

- SQL (Structured Query Language, 結構化查詢語言) 是一種用來與關連式資料庫系統對話而使用的標準語言，由IBM於1970年代所研發出來的，目前所有市場的資料庫管理系統幾乎都支援SQL。
- 美國國家標準協會(ANSI)與國際標準組織(ISO)於1987年認定SQL的標準版本(SQL/87或稱SQL/1)。
- SQL語法的版本持續演進中…
  - SQL-87, SQL-92, SQL:1999, SQL:2003, SQL:2006, SQL:2008, SQL:2011

- **SQL指令的種類：**
  - **資料定義語言 (Data Definition Language, DDL)**
    - 用來宣告(或建立)資料庫物件
    - 針對Table, View或Database做建立(Create)、刪除(Drop)、更改(Alter)等動作。
  - **資料處理語言 (Data Manipulation Language, DML)**
    - 用來操作資料庫中的資料
    - 針對Table內的Data，做插入(Insert)、更新(Update)、刪除(Delete)等動作。
  - **資料控制語言 (Data Control Language, DCL)**
    - 用來從事資料庫的權限控管，如Grant、Revoke、Alter Password等動作。
  - **資料查詢語言 (Data Query Language, DQL)**
    - 用來查詢資料庫中的資料 (某些書將DQL併入DML一起討論)
  - **資料管理指令 (Data Administration Commands)**
    - 用來從事資料庫的稽核與分析
  - **交易控制指令 (Transactional Control Commands)**
    - 用來管理資料庫的交易動作

- 市面上所有商用資料庫系統皆支援ANSI SQL語法。
- 由於各廠商為突顯自家資料庫系統之獨特性，會對部份的ANSI SQL語法做些微修改，以支援自家資料庫系統所發展之各項功能。
- 若需操作不同資料庫系統，建議準備有介紹各家資料庫系統SQL語法之辭典，以應付不時之需。



SQL資料型態

- 字串 (Character Strings)
  - CHAR(n)：固定長度字元串 (n為字元個數)
  - VARCHAR(n)：變動長度字元串
  - BIT(n)：固定長度位元串 (n為位元個數)
  - BIT VARING(n)：變動長度位元串
- MySQL的字串型態：

字串型態	大小	意義
CHAR(M)	M BYTES	M 可以是 1 到 255 的長度
VARCHAR(M)	M BYTES	不定長度的字元，使用 1 到 255BYTES
TINYBLOB,TINYTEXT	L+1 BYTES	不定長度的字元
MEDIUMBLOB,MEDIUMTEXT	L+3 BYTES	不定長度字元
BLOB，TEXT	L+2 BYTES	不定長度，最多到 65535 字元
LONGBLOB,LONGTEXT	L+4 BYTES	不定長度，最多到 4294967295
ENUM("VALUE","VALUE2",...)	1 或 2BYTES	列舉型態
SET	1,2,3,4,8 BYTES	集合型態

- 數值串(Numeric Strings)
  - INT, INTEGER：整數
  - DEC(m,n), DECIMAL(m,n), NUMERIC(m,n)：格式化數值 (m：總位數或精確度，n：小數位數)
  - SMALLINT：短整數
  - FLOAT：浮點數
  - REAL：單精度實數 (32bits)
  - DOUBLE PRECISION：雙精度實數 (64bits)

- **MySQL的數值串型態：**

名稱	意義
INT	整數 範圍：包含負數從-2147683648 到 2147483648 正整數從 0 到 4294967295
FLOAT FLOAT(4) FLOAT(8)	浮點數(包含小數部份的數字) 浮點數使用 4 個 BYTES 浮點數使用 8 個 BYTES
TINYINT	非常小的整數，使用 1 個 BYTE。 範圍：包含負號從-128 到 127 正整數從 0 到 255
SMALLINT	小的整數
MEDIUMINT	中型的整數
BIGINT	大的整數，使用 8 個 BYTES。從 -9223372036854775808 到 9223372036854774807
DOUBLE	倍精度的浮點數，使用 8 個 BYTES。
DECIMAL	字串形式表示的浮點數

- 日期/時間 (Date/Time)資料型態
  - DATE：一般格式為 YYYY-MM-DD
  - TIME：一般格式為 HH:MM:SS
  - TIMESTAMP：時間戳記，由DATE+TIME+六位以上小數秒數
    - 用以記錄交易進入系統的時間順序
  - INTERVAL：時間區間
- MySQL的日期/時間型態：

資料型態	意義
DATE	日期以"YYYY-MM-DD"使用 3 個 BYTES，包含日期從西元 1000 年 1 月 1 日(1000-01-01)到西元 9999 年 12 月 31 日止(9999-12-31)。
DATETIME	從西元 1000 年 1 月 1 日 0 時 0 分 0 秒(1000-01-01 00:00:00)開始到西元 9999 年 12 月 31 日 23 時 59 分 59 秒(9999-12-331 23:59;59)
TIME	時間以"HH:MM:SS"格式表示，使用 3 個 BYTES。
DATETIME	日期與時間以"YYYY-MM-DD HH:MM:SS"格式表示
TIMESTAMP	時間紀錄從西元 1970 年 1 月 1 日 0 時 0 分 0 秒開始，以 YYYYMMDDHHMMSS 格式表示，使用 4 個 BYTES。
YEAR	以 YYYY 格式表示，使用 1 個 BYTES。

Data Definition Language, DDL (資料定義語言) - (1)

- DDL主要有CREATE, DROP, ALTER三個指令，並可針對以下三個資料庫物件進行操作：
  - 資料庫 (database，或稱Schema)
  - 表格 (Table)
  - *景觀 (View)*

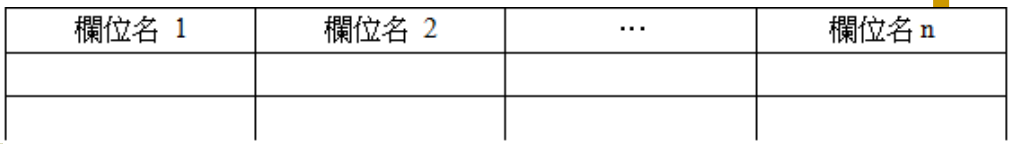
建立、刪除資料庫

- 在進行Create、Drop資料庫的指令操作時，對 “資料庫” 這個關鍵字有時是用 “Schema” 來表示。
  - CREATE DATABASE/SCHEMA: 建立一個新的DB
    - CREATE SCHEMA <資料庫名>; 或 CREATE DATABASE <資料庫名>;
    - 例：CREATE SCHEMA Jacy_Database;

      CREATE DATABASE Jacy_Database;
  - DROP SCHEMA/DATABASE: 刪除一個DB
    - DROP SCHEMA <資料庫名>; 或 DROP DATABASE <資料庫名>;
    - 例：DROP SCHEMA Jacy_Database;

      DROP DATABASE Jacy_Database;
- 上述語法在MySQL中亦適用，且指令不區分大小寫。

建立、刪除、更改表格



- **CREATE TABLE: 建立一個新的關聯表格**
  - CREATE TABLE <table name>(

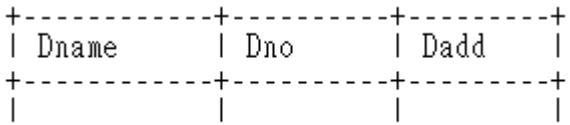
```
<欄位名 1> <data type> [Null/Not null] [DEFAULT <預設值>],
<欄位名 2> <data type> [Null/Not null] [DEFAULT <預設值>],
                        ⋮
<欄位名 n> <data type> [Null/Not null] [DEFAULT <預設值>],
PRIMARY [KEY](<欄位名>),
UNIQUE [KEY](<欄位名>),
FOREIGN KEY(<欄位名>) REFERENCES <表格名(欄位名)> [ON
 DELETE…/ON UPDATE…]
);
```

指定表格中的欄位

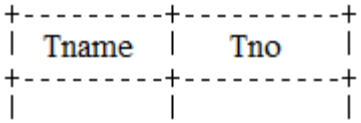
對具有特定用途/限制之欄位加以設定

括號內最後一行指令不需要逗號!!

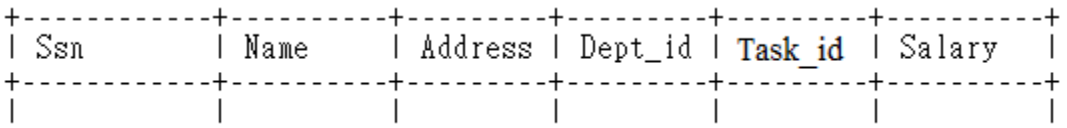
- CREATE TABLE Department
  (Dname CHAR(10) NOT NULL,
  Dno INT,
  Dadd CHAR(20),
  PRIMARY KEY(Dno)
  );



- CREATE TABLE Task
  (Tname CHAR(10),
  Tno INT NOT NULL,
  PRIMARY KEY(Tno)
  );



- CREATE TABLE Employee
  (Ssn CHAR(10),
    Name CHAR(10),
    Address VARCHAR(50),
    Dept_id INT,
    Task_id INT,
    Salary NUMERIC(8,1) NOT NULL DEFAULT 18000,
    PRIMARY KEY(Ssn),
    UNIQUE(Name),
    FOREIGN KEY(Task_id) REFERENCES Task(Tno),
    FOREIGN KEY(Dept_id) REFERENCES Department(Dno) ON DELETE Cascade
  );



表格 A

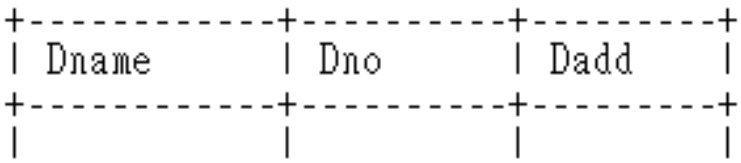
員工代號	姓名	部門代號
A01	張三	D002
A02	李四	D003
A03	王五	D002

表格 B

部門代號	部門名稱	所在地
D001	人事部	台中
D002	工務部	中壢
D003	資訊室	台北

- **註：設定外來鍵時，關於ON DELETE與ON UPDATE的處理動作有以下幾種：**
  - **RESTRICT (No Action)：發生違反完整性限制的操作時，DBMS不允許該操作執行**
  - **CASCADE：發生違反參考完整性限制的操作時，外來鍵內的資料連帶更新或刪除**
  - **SET NULL：發生違反參考完整性限制的操作時，外來鍵內的資料設為空值**

- **欄位限制之設定有兩種：**
  - **Table-level Constraint**
    - **CREATE TABLE Department**
      (Dname CHAR(10),
      Dno INT,
      Dadd CHAR(20),
      PRIMARY KEY(Dno),
      Unique(Dname)
      );
  - **Column-level Constraint**
    - **CREATE TABLE Department**
      (Dname CHAR(10) Unique,
      Dno INT PRIMARY KEY,
      Dadd CHAR(20)
      );



通常建構複雜的表格時，兩者會混用

- 關於MySQL資料表格有兩種常用的類型：
  - MyISAM
    - 為MySQL預設的表格類型
    - 成熟、穩定、容易管理。若無特殊需求，應以此類型為主。
  - InnoDB
    - 支援交易(Transaction)機制、外來鍵(Foreign Key)、當機復原(若系統未遭受損壞時使用)
- MyISAM v.s. InnoDB
  - 若想追求使用空間與執行效率，建議採用MyISAM
  - 若著重交易工作、安全性考量、有外來鍵的使用需求或是可能有多人同時修改資料的情況，則建議採用InnoDB

- **在MySQL中，建立一個新的表格：**
  - **CREATE TABLE <table name> (**

    <attribute name 1> <data type> <(not) null> <default value>,

    <attribute name 2> <data type> <(not) null> <default value>,

    ⋮

    <attribute name n> <data type> <(not) null> <default value>,

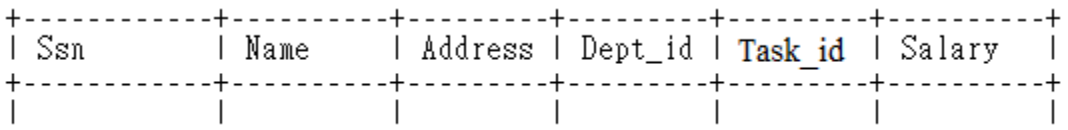
    PRIMARY KEY(<attribute name>),

    UNIQUE(<attribute name>),

    FOREIGN KEY(<attribute name>) REFERENCES <表格名(欄位名)>
    [ON DELETE…/ON UPDATE…]

) ENGINE = MyISAM/InnoDB; (此行若不打，則預設為MyISAM格式)
- **於MySQL刪除、更改一個表格，可採用後面即將介紹的SQL標準語法。**

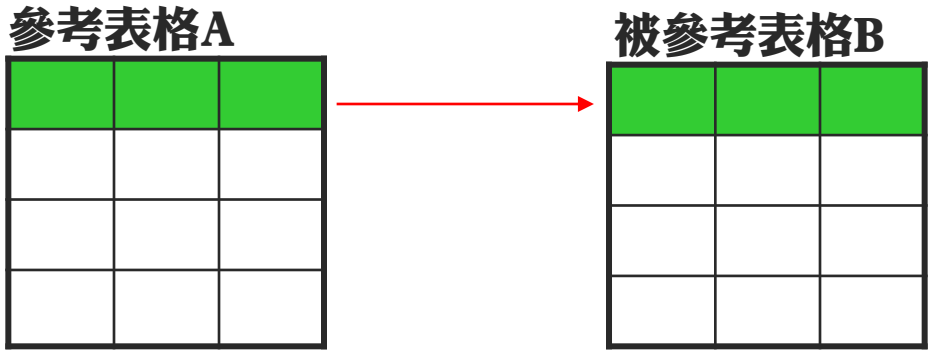
```
CREATE TABLE Employee_01
  ( Ssn CHAR(10) NOT NULL,
    Name CHAR(10) NOT NULL,
    Address VARCHAR(50),
    Dept_id INT not null,
    Task_id INT not null,
    Salary NUMERIC(8,1) NOT NULL DEFAULT 18000,
    Primary Key(Ssn)
  ) ENGINE = InnoDB;
```



- DROP TABLE: 刪除一個關聯 (表格)

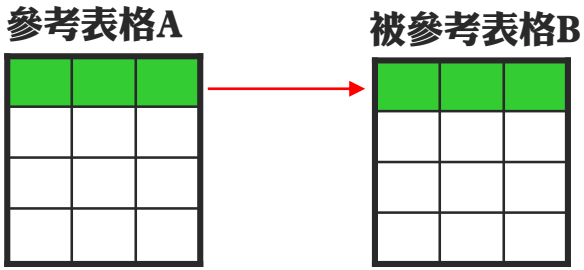
  DROP TABLE <表格名>;

  - 例：DROP TABLE Employee_01;
- 注意：當要刪除掉彼此有參考關係的表格時，先刪除參考表格，再刪除被參考表格。



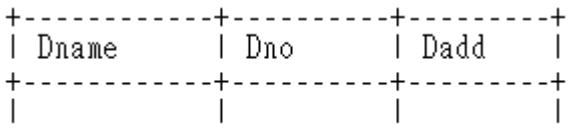
若表格間有外來鍵存在時之建表、刪表順序

- **建表順序：**
  - 先建立“被參考表格”(即：Task, Department)，再建立“參考表格”(∵要讓外來鍵有對象可以參考)
  - 在MySQL中，不論是參考表格，還是被參考表格，只要與外來鍵設定有關的表格，皆需設定成InnoDB類型的表格，參考完整性限制方可正常執行。

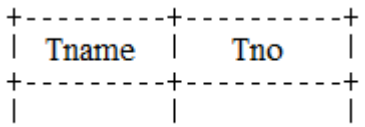


- **刪表順序：**
  - 先刪除參考表格，再刪除被參考表格。(∵要讓外來鍵有對象可以參考)

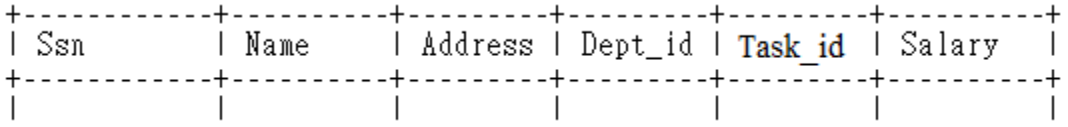
- 假設有三個表格需要建立：Department, Task, Employee。其中：
  - Employee有兩個外來鍵，分別指向Department與Task
- CREATE TABLE Department
  (Dname CHAR(10) NOT NULL,
  Dno INT NOT NULL,
  Dadd CHAR(20),
  PRIMARY KEY(Dno)
) ENGINE=INNODB;



- CREATE TABLE Task
  (Tname CHAR(10) NOT NULL,
  Tno INT NOT NULL,
  PRIMARY KEY(Tno)
) ENGINE=INNODB;



```
CREATE TABLE Employee(
  Ssn CHAR(10) NOT NULL,
  Name CHAR(10) NOT NULL,
  Address VARCHAR(50),
  Dept_id INT,
  Task_id INT,
  Salary NUMERIC(8,1) NOT NULL DEFAULT 18000,
  PRIMARY KEY(Ssn),
  FOREIGN KEY(Task_id) REFERENCES Task(Tno),
  FOREIGN KEY(Dept_id) REFERENCES Department(Dno) ON
  Delete CASCADE
) ENGINE=INNODB;
```



- ALTER TABLE: 更改一個關聯 (表格) 中之某欄位的基本定義與限制。包括：
  - 增加(ADD)欄位、刪除(DROP)欄位
  - 修改(ALTER/CHANGE/MODIFY…) 欄位預設值、定義、或名稱
  - 更改表格名稱(RENAME)
  - 更改表格類型(ENGINE)

  ALTER TABLE <表格名>

  ADD/DROP/ALTER/CHANGE/MODIFY/ENGINE/RENAME

- **增加欄位：**
  - ALTER TABLE <表格名> ADD <新欄位名> <data type> [Null/Not null] [DEFAULT <預設值>];
  - 例：ALTER TABLE Emp
    ADD Sex CHAR(1);
- **刪除欄位：**
  - ALTER TABLE <表格名> DROP <欄位名> [RESTRICT/CASCADE];
  - 例：ALTER TABLE Emp
    DROP Sex;

- **修改欄位 1 (增改/刪除預設值)：**
  - ALTER TABLE <表格名> ALTER <欄位名> DROP DEFAULT;
  - ALTER TABLE <表格名> ALTER <欄位名> SET DEFAULT <預設值>;
  - 例1：ALTER TABLE Emp

    ALTER Salary DROP DEFAULT;
  - 例2：ALTER TABLE Emp

    ALTER Salary SET DEFAULT 18000;
- **修改欄位 2 (更改欄位定義，不含修改欄位名)：**
  - ALTER TABLE <表格名> MODIFY <欄位名> <data type> [Null/Not null];
  - Null/Not null若不設定，則預設為Null
  - 例：ALTER TABLE Emp

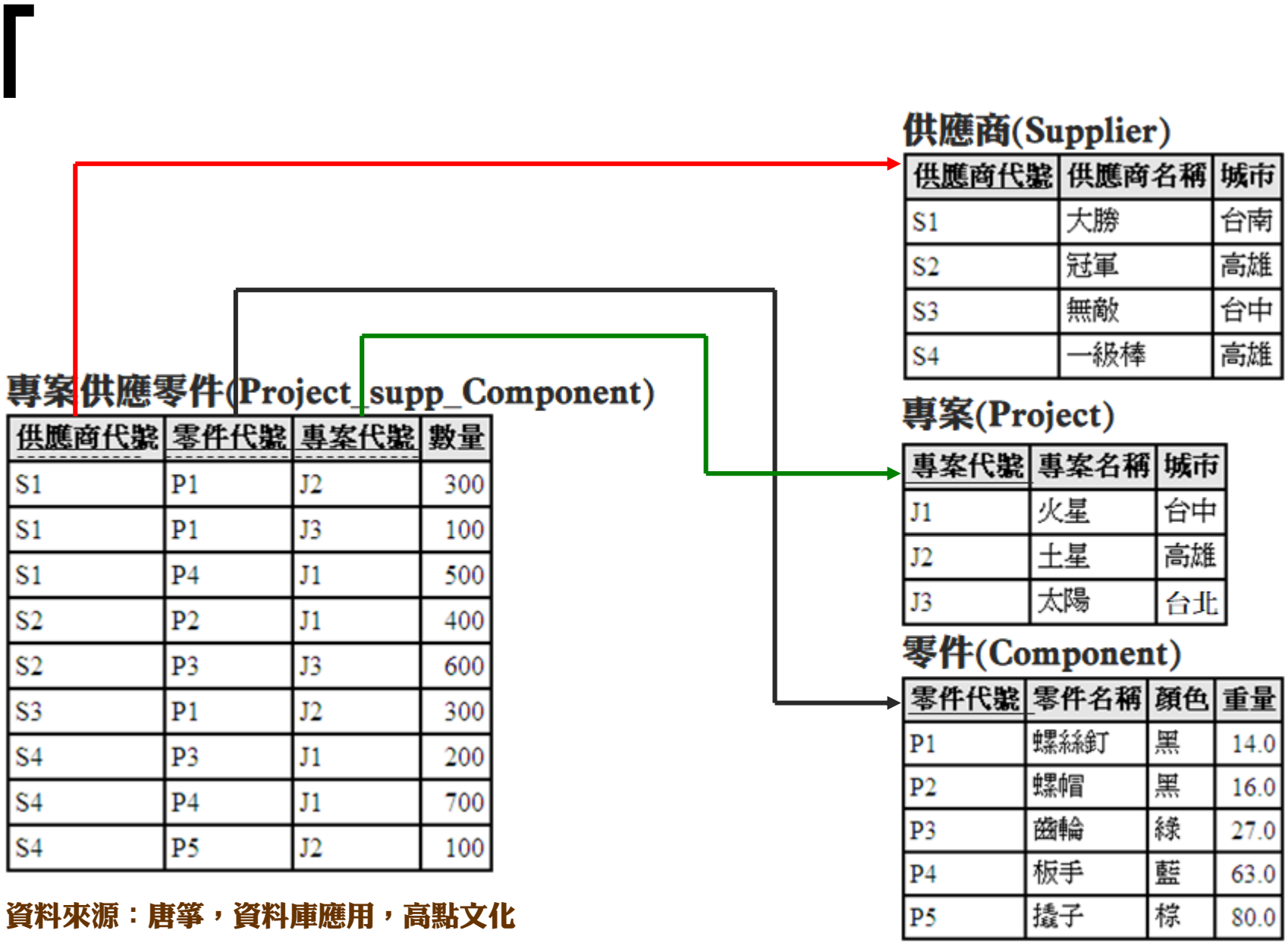
    MODIFY Salary decimal(8,1) Not null;

- 修改欄位 3 (更改欄位定義，含修改欄位名)：
  - ALTER TABLE <表格名> CHANGE <舊欄位名> <新欄位名> <data type> [Null/Not null];
  - Null/Not null若不設定，則預設為Null
  - 例：ALTER TABLE Emp
    CHANGE Salary Sal decimal(8,1);
- 更改表格類型：
  - ALTER TABLE <表格名> ENGINE <類型> ;
  - 例：ALTER TABLE Emp
    ENGINE InnoDB;
- 更改表格名稱：
  - ALTER TABLE <舊表格名> RENAME <新表格名> ;
  - 例：ALTER TABLE Emp
    RENAME Emp_01;

※練習範例※

- 請建立slide 11的四個表格。令：
  - Supplier
    - 供應商代號 CHAR(4)
    - 供應商名稱 CHAR(10)
    - 城市 CHAR(6)
  - Project
    - 專案代號 CHAR(4)
    - 專案名稱 CHAR(10)
    - 城市 CHAR(6)
  - Component
    - 零件代號 CHAR(4)
    - 零件名稱 CHAR(10)
    - 顏色 CHAR(4)
    - 重量 INT
  - Project_supp_Component
    - 供應商代號 CHAR(4)
    - 專案代號 CHAR(4)
    - 零件代號 CHAR(4)
    - 數量 INT

註：外來鍵皆設成 ON Delete CASCADE。表格類型皆為InnoDB。



供應商(Supplier)

供應商代號	供應商名稱	城市
S1	大勝	台南
S2	冠軍	高雄
S3	無敵	台中
S4	一級棒	高雄

專案供應零件(Project_supp_Component)

供應商代號	零件代號	專案代號	數量
S1	P1	J2	300
S1	P1	J3	100
S1	P4	J1	500
S2	P2	J1	400
S2	P3	J3	600
S3	P1	J2	300
S4	P3	J1	200
S4	P4	J1	700
S4	P5	J2	100

專案(Project)

專案代號	專案名稱	城市
J1	火星	台中
J2	土星	高雄
J3	太陽	台北

零件(Component)

零件代號	零件名稱	顏色	重量
P1	螺絲釘	黑	14.0
P2	螺帽	黑	16.0
P3	齒輪	綠	27.0
P4	板手	藍	63.0
P5	撬子	棕	80.0

資料來源：唐筆，資料庫應用，高點文化

Data Manipulation Language, DML (資料處理語言)

- DML是針對關聯中的資料部份從事處理，包含Insert, Update, Delete指令。
  - INSERT: 插入一筆新的紀錄到關聯中。
  - DELETE:根據WHERE條件刪除關聯中的紀錄。
  - UPDATE: 根據WHERE條件更改關聯中的屬性值。

- **INSERT: 插入一筆新的紀錄到關聯中。**

  INSERT INTO <table name>[(attribute1, attribute2,…)] VALUES (…)

  - 例：假設現有一關聯表格如下：

    employee(Ssn, Name, Address, Dept_id, Proj_id, Salary)

    - 範例1：此例為插入部份欄位資料

      INSERT INTO employee (Ssn, Name, Address, Salary)
      VALUES ('F111111111', '操勞哥', '苗栗', 20000);

    - 範例2：此例為插入全部欄位資料

      INSERT INTO employee
      VALUES ('F111111112', '打混王', '台北', 021, 035, 25000);

  - 若多個表格間有外來鍵時，先插入 "被參考表格" 的資料，再插入 "參考表格" 的資料

Ssn	Name	Address	Dept_id	Proj_id	Salary

- **DELETE: 根據WHERE條件刪除關聯中的紀錄。**

  DELETE FROM <table name> WHERE <condition>

  - **例：假設現有一關聯表格employee(Ssn, Name, Address, Dept_id, Proj_id, Salary)**
    - **DELETE FROM employee WHERE Salary < 20000;**
    - **DELETE FROM employee WHERE Ssn = 'F111111111';**
    - **DELETE FROM employee;**

- **UPDATE: 根據WHERE條件更改關聯中的屬性值。**

  UPDATE <表格名>

  SET <屬性名>=<new value> [, <屬性名 2>=<new value 2>]

  WHERE <條件>

  - 例：假設現有一關聯表格employee(Ssn, Name, Address, Dept_id, Proj_id, Salary)

    ```
    UPDATE employee
    SET Address = '台東', Salary =10000
    WHERE Ssn = 'F111111113';
    ```

※練習範例※

- 請為slide 9所建立的四個空表格插入其應有的資料。

Data Query Language, DQL (資料查詢語言)

- DQL用以查詢資料庫的相關資料，語法如下：

  SELECT <attribute list>

  FROM <table list>

  WHERE <condition>

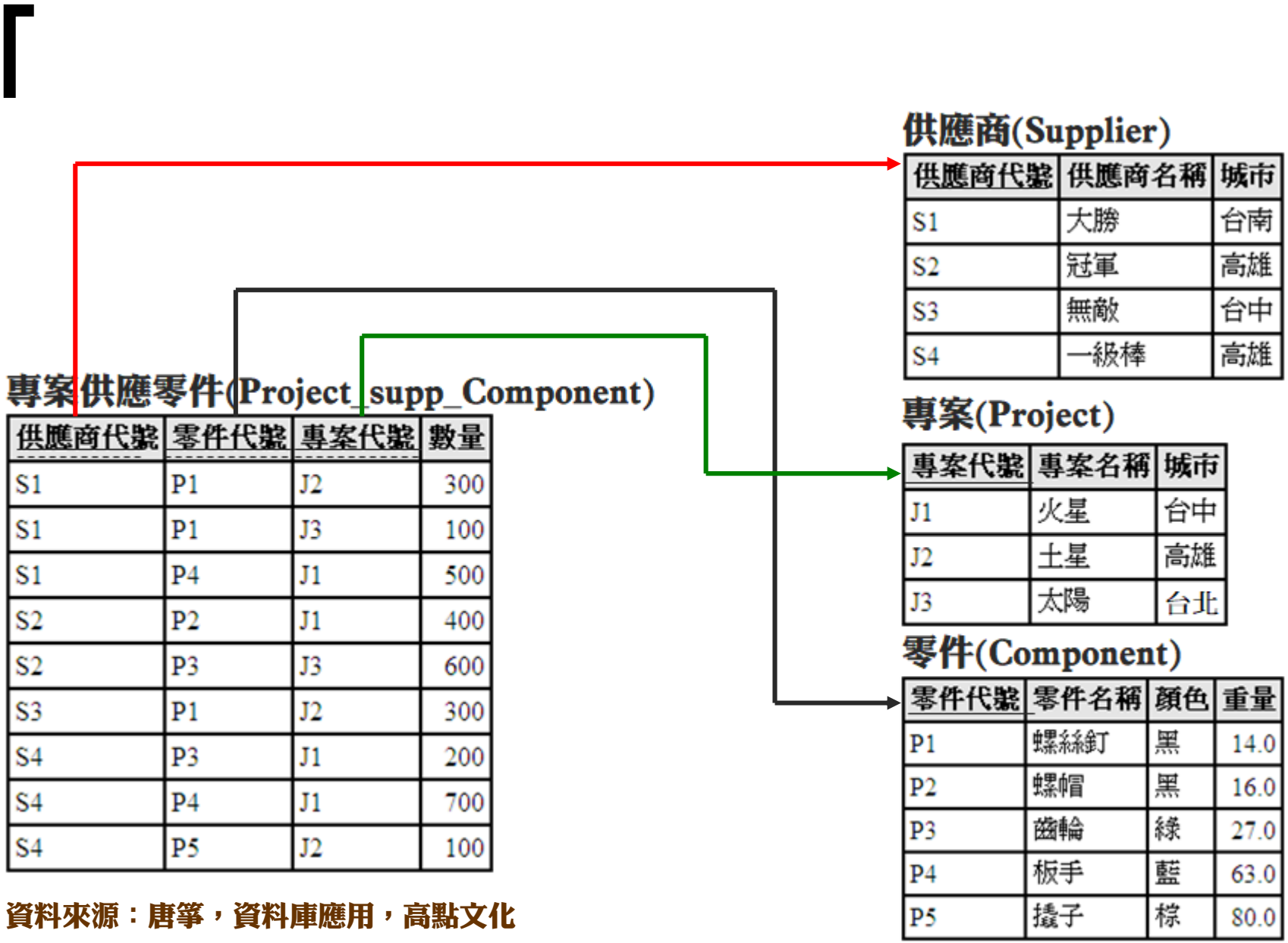
  GROUP BY <grouping attributes>

  HAVING <grouping condition>

  ORDER BY <column name> ASC/DESC

- 執行順序：
  - FROM→WHERE →GROUP BY →HAVING →SELECT →ORDER BY

- 各個子句的說明：
  - SELECT: 指定查詢所欲輸出的欄位
  - FROM: 指定查詢所牽涉到的表格
  - WHERE: 指定查詢的限制條件
  - GROUP BY: 將查詢資料依照某個指定的欄位加以分群
  - HAVING: GROUP BY的限制條件，必須配合GROUP BY使用
  - ORDER BY: 依照某屬性值作遞增(ASE)或遞減(DESC)排序 (預設為ASC)



供應商(Supplier)

供應商代號	供應商名稱	城市
S1	大勝	台南
S2	冠軍	高雄
S3	無敵	台中
S4	一級棒	高雄

專案供應零件(Project_supp_Component)

供應商代號	零件代號	專案代號	數量
S1	P1	J2	300
S1	P1	J3	100
S1	P4	J1	500
S2	P2	J1	400
S2	P3	J3	600
S3	P1	J2	300
S4	P3	J1	200
S4	P4	J1	700
S4	P5	J2	100

專案(Project)

專案代號	專案名稱	城市
J1	火星	台中
J2	土星	高雄
J3	太陽	台北

零件(Component)

零件代號	零件名稱	顏色	重量
P1	螺絲釘	黑	14.0
P2	螺帽	黑	16.0
P3	齒輪	綠	27.0
P4	板手	藍	63.0
P5	撬子	棕	80.0

資料來源：唐筆，資料庫應用，高點文化

- SQL查詢語言的六個子句中，只有Select和From是必要的，其它的依需求而定。
- 例：將表格Supplier中，所有供應商的資料全部列出

```
SELECT *
FROM Supplier;
```

  - SELECT * ：將FROM中關聯表格的所有屬性全部顯示
  - 查詢中沒有WHERE子句，表示沒有限制條件，即表格的所有資料紀錄皆會全部列出
- 例：將表格Supplier中，位於台南的所有供應商資料列出

```
SELECT *
FROM Supplier
WHERE 城市 = '台南' ;
```

供應商(Supplier)

供應商代號	供應商名稱	城市
S1	大勝	台南
S2	冠軍	高雄
S3	無敵	台中
S4	一級棒	高雄

SQL語法的運算子

- SQL語法中可能會用到的運算子：

運算子	意義	範例
+	加	5+5
-	減	8-2
*	乘	6*5
/	除	6/3
%	取餘數	5%2

運算子	意義	範例
and	A 和 B 同時爲真則爲真	真 and 真傳回真
or	A 或 B 只要一個爲真就爲真	真 or 假 傳回真
&&	A 和 b 同時爲真則爲真	真&&真 傳回真
||	A 或 B 只要一個爲真就爲真	真||假 傳回真

運算子	意義	範例
>	大於	5 > 3 傳回真
<	小於	6 < 8 傳回真
=	等於	6 = 6 傳回真
!=	不等餘	3 !=5 傳回真

※範例題組1※

- 依照slide 37的四個表格，用SQL回答下列問題：
  - 列出所有供應商名稱。

Sol:

```
SELECT 供應商名稱
FROM Supplier;
```

供應商(Supplier)

供應商代號	供應商名稱	城市
S1	大勝	台南
S2	冠軍	高雄
S3	無敵	台中
S4	一級棒	高雄

  - 列出所有重量在20以上，且不為黑色的零件名稱、顏色、重量。

(學習重點：用到不同的運算子)

Sol:

```
SELECT 零件名稱, 顏色, 重量
FROM Component
WHERE 重量 >=20 AND 顏色!='黑';
```

零件(Component)

零件代號	零件名稱	顏色	重量
P1	螺絲釘	黑	14.0
P2	螺帽	黑	16.0
P3	齒輪	綠	27.0
P4	板手	藍	63.0
P5	撬子	棕	80.0

欄位名稱重覆處理

- 複雜的查詢任務可能需要多個來源表格，而使用這些表格時，可能會發生**位於不同表格中之同名欄位被放在一起使用**的情況。
- 若沒有明確指示，資料庫系統不知道這些同名欄位是隸屬於哪一個表格。因此，需進行**欄位名稱重覆處理**。
- **作法**：這些同名欄位之名稱在使用時，需加上**表格名稱**：

  table_name.attribute_name

  - 如：專案.城市，供應商.城市

專案(Project)

專案代號	專案名稱	城市
J1	火星	台中
J2	土星	高雄
J3	太陽	台北

供應商(Supplier)

供應商代號	供應商名稱	城市
S1	大勝	台南
S2	冠軍	高雄
S3	無敵	台中
S4	一級棒	高雄

專案(Project)

專案代號	專案名稱	城市
J1	火星	台中
J2	土星	高雄
J3	太陽	台北

零件(Component)

零件代號	零件名稱	顏色	重量
P1	螺絲釘	黑	14.0
P2	螺帽	黑	16.0
P3	齒輪	綠	27.0
P4	板手	藍	63.0
P5	撬子	棕	80.0

※範例題組2※

- 依照slide 37的四個表格，用SQL回答下列問題：
  - 依照數量由小到大列出供應商S1所參與之專案名稱、零件名稱，以及數量。

(學習重點：1. 排序子句 ORDER BY;
2. 多個表格的欄位名稱重覆之處理)

專案供應零件(Project_supp_Component)

供應商代號	零件代號	專案代號	數量
S1	P1	J2	300
S1	P1	J3	100
S1	P4	J1	500
S2	P2	J1	400
S2	P3	J3	600
S3	P1	J2	300
S4	P3	J1	200
S4	P4	J1	700
S4	P5	J2	100

Sol:

```
SELECT 專案名稱, 零件名稱, 數量
FROM Project, Component, Project_supp_Component
WHERE Project.專案代號= Project_supp_Component.專案代號 AND
      Component. 零件代號= Project_supp_Component.零件代號 AND
      Project_supp_Component.供應商代號='S1'
ORDER BY 數量;
```

SQL語法的函數

- 聚合函數：
  - COUNT(attribute_name): 計算非空屬性值個數
  - SUM(attribute_name): 計算屬性中數值的總合
  - AVG(attribute_name): 計算屬性中數值的平均
  - MAX(attribute_name): 找出屬性中數值的最大值
  - MIN(attribute_name): 找出屬性中數值的最小值

(其它類型的SQL函數請參考網路講義)

http://www.codedata.com.tw/database/mysql-tutorial-4-expression-function/

※範例題組3※

- 由零件表格中，統計所有零件之總重量。

  (學習重點：聚合函數的應用)

  Sol:

  ```
  SELECT SUM(重量)
   FROM Component;
  ```

零件(Component)

零件代號	零件名稱	顏色	重量
P1	螺絲釘	黑	14.0
P2	螺帽	黑	16.0
P3	齒輪	綠	27.0
P4	板手	藍	63.0
P5	撬子	棕	80.0

- 由專案供應零件中，統計有支援專案執行之各零件的總數量並列出零件代號。

  (學習重點：分群子句 GROUP BY)

  Sol:

  ```
  SELECT 零件代號, SUM(數量)
   FROM Project_supp_Component
   GROUP BY 零件代號;
  ```

專案供應零件(Project_supp_Component)

供應商代號	零件代號	專案代號	數量
S1	P1	J2	300
S1	P1	J3	100
S1	P4	J1	500
S2	P2	J1	400
S2	P3	J3	600
S3	P1	J2	300
S4	P3	J1	200
S4	P4	J1	700
S4	P5	J2	100

- 由專案供應零件表，統計有支援專案執行之各零件中，總數量超過1000之零件的總數量，並列出其零件代號。

  (學習重點：1. GROUP BY專用的限制條件子句 HAVING; 2. 對欄位取別名)

  Sol:

```
SELECT 零件代號, SUM(數量)
 FROM Project_supp_Component
 GROUP BY 零件代號
 HAVING SUM(數量)>1000;
```

專案供應零件(Project_supp_Component)

供應商代號	零件代號	專案代號	數量
S1	P1	J2	300
S1	P1	J3	100
S1	P4	J1	500
S2	P2	J1	400
S2	P3	J3	600
S3	P1	J2	300
S4	P3	J1	200
S4	P4	J1	700
S4	P5	J2	100

- 【注意】利用聚合函數所設立之條件，不能直接寫在WHERE子句
  - Where子句是繼From子句後，第二個會被執行到的SQL查詢語句
  - 若無搭配Group By先做分群，則聚合函數僅會得到單一筆計算結果(如：範例題組3的第一題)；若對此計算結果再做條件過濾是會有函數使用上的問題!!

欄位與表格之別名設定

- 可以對使用到的關聯表格或是欄位取別名(Alias)：
  - <原表格名稱> AS <別名> 或是 <原表格名稱> <別名>
  - <原欄位名稱> AS <別名> 或是 <原欄位名稱> <別名>

※範例題組4※

- 列出供應商 “大勝” 有供應的所有零件中，重量最重之重量為何。

  (學習重點：1. 聚合函數MAX; 2. 對表格取別名; 3. 對欄位取別名)

Sol:

```
SELECT MAX(重量) AS 最大重量
FROM Supplier as S, Component as C, Project_supp_Component P
WHERE S.供應商代號= P.供應商代號 AND
      C. 零件代號= P.零件代號 AND
      S.供應商名稱='大勝' ;
```

專案供應零件(Project_supp_Component)

供應商代號	零件代號	專案代號	數量
S1	P1	J2	300
S1	P1	J3	100
S1	P4	J1	500
S2	P2	J1	400
S2	P3	J3	600
S3	P1	J2	300
S4	P3	J1	200
S4	P4	J1	700
S4	P5	J2	100

零件(Component)

零件代號	零件名稱	顏色	重量
P1	螺絲釘	黑	14.0
P2	螺帽	黑	16.0
P3	齒輪	綠	27.0
P4	板手	藍	63.0
P5	撬子	棕	80.0

供應商(Supplier)

供應商代號	供應商名稱	城市
S1	大勝	台南
S2	冠軍	高雄
S3	無敵	台中
S4	一級棒	高雄

- **聚合函數中COUNT的用法：**
  - **COUNT(*): 計算表格中共有幾筆非空記錄**
  - **COUNT(欄位名稱): 此屬性有幾筆非空值**
  - **COUNT(DISTINCT 欄位名稱): 此屬性有幾筆不同的非空值**

供應商(Supplier)

供應商代號	供應商名稱	城市
S1	大勝	台南
S2	冠軍	高雄
S3	無敵	NULL
S4	一級棒	高雄

※範例題組5※

- 若表格Supplier加入下列資料：

```
INSERT INTO Supplier(供應商代號,供應商名稱)
VALUES ('S5', '歐羅肥');
```

- 請執行並區分以下三個SQL指令
  (學習重點： 聚合函數COUNT；對欄位取別名)

1. SELECT COUNT(*) as 供應商數目
   FROM Supplier;
2. SELECT COUNT(城市)
   FROM Supplier;
3. SELECT COUNT(DISTINCT 城市)
   FROM Supplier;

供應商(Supplier)

供應商代號	供應商名稱	城市
S1	大勝	台南
S2	冠軍	高雄
S3	無敵	台中
S4	一級棒	高雄
S5	歐羅肥	NULL

巢狀SQL查詢

- 查詢中又包含另一個查詢。
- 問題：有哪些員工的Salary高於員工編號為F111111115的員工之Salary?

Main query (主查詢):

哪些員工的Salary高於員工編號為F111111115的員工之Salary?

Subquery (子查詢):

F111111115的員工之Salary為何?

Sol:

```
SELECT Ssn, Emp_id, Salary
FROM Employee
WHERE Salary > (SELECT Salary
                FROM Employee
                WHERE Ssn='F111111115');
```

- 巢狀查詢的語法

```
SELECT    欄位名
FROM      表格名
WHERE     expr operator
                (SELECT   欄位名
                 FROM     表格名);
```

- 在實作上，子查詢不會僅出現在WHERE子句後!!
  - 巢狀查詢語法的變化非常多，在此不多做介紹。但是請記住一個概念：子查詢的回傳結果也是一個二維表格!!!

巢狀查詢的分類

- 根據Subquery回傳結果的筆數，巢狀查詢可分成：
  - Single-row Subquery
  - Multi-row Subquery
- 根據Subquery是否可以獨立執行，巢狀查詢可分成：
  - 標準子查詢：子查詢可以獨立運作
  - 關聯子查詢：子查詢會使用到主查詢的表格，無法獨立運作
- 前述的範例為Single-row的標準子查詢。
  - 其Subquery僅回傳單一筆的資料，且子查詢可以獨立運作

巢狀查詢運算子 (Operator)

- **Single-row Subquery**

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to

- **Multi-row Subquery**
  - **IN, ANY, ALL**

- **IN: 判斷 IN 之前的待比較資料，是否存在於IN後面子查詢所回傳的查詢結果集合中。**
  - **NOT IN則為IN的反義**
- **ALL, ANY：用來與子查詢所回傳的結果集合做數值的比較 (通常會配合 “<”、 “>” 或 “NOT” 等運算子來用)。**
  - **ALL：當ALL之前的待比較資料，滿足子查詢所回傳的所有查詢結果即成立。**
  - **ANY：當ANY之前的待比較資料，滿足子查詢所回傳的任一查詢結果即成立。**

※範例題組6※

供應商(Supplier)

供應商代號	供應商名稱	城市
S1	大勝	台南
S2	冠軍	高雄
S3	無敵	台中
S4	一級棒	高雄

零件(Component)

零件代號	零件名稱	顏色	重量
P1	螺絲釘	黑	14.0
P2	螺帽	黑	16.0
P3	齒輪	綠	27.0
P4	板手	藍	63.0
P5	搖子	棕	80.0

(學習重點： 巢狀查詢運算子; Multi-row Subquery; 標準子查詢)

專案供應零件(Project_supp_Component)

供應商代號	零件代號	專案代號	數量
S1	P1	J2	300
S1	P1	J3	100
S1	P4	J1	500
S2	P2	J1	400
S2	P3	J3	600
S3	P1	J2	300
S4	P3	J1	200
S4	P4	J1	700
S4	P5	J2	100

- 列出供應商“大勝”有供應給專案使用的零件之平均重量

  Sol:

```
SELECT avg(Component.重量)
FROM Component
WHERE Component.零件代號 IN
        (SELECT Distinct PSC.零件代號
         FROM Supplier AS S, Project_supp_Component AS PSC
         WHERE S.供應商代號=PSC.供應商代號 AND
               S.供應商名稱='大勝');
```

- 列出重量大於供應商代號S2有供應給專案使用之所有零件的零件名稱

  Sol:

```
SELECT 零件名稱
FROM Component
WHERE 重量 >ALL (SELECT C.重量
                 FROM Project_supp_Component AS PSC, Component AS C
                 WHERE PSC.零件代號=C.零件代號 AND
                       PSC.供應商代號='S2');
```

- 列出重量大於供應商代號S2有供應給專案使用之任一零件的零件名稱

  Sol:

```
SELECT 零件名稱
FROM Component
WHERE 重量 >ANY (SELECT C.重量
                 FROM Project_supp_Component AS PSC,
                      Component AS C
                 WHERE PSC.零件代號=C.零件代號 AND
                       PSC.供應商代號='S2');
```

專案供應零件(Project_supp_Component)

供應商代號	零件代號	專案代號	數量
S1	P1	J2	300
S1	P1	J3	100
S1	P4	J1	500
S2	P2	J1	400
S2	P3	J3	600
S3	P1	J2	300
S4	P3	J1	200
S4	P4	J1	700
S4	P5	J2	100

零件(Component)

零件代號	零件名稱	顏色	重量
P1	螺絲釘	黑	14.0
P2	螺帽	黑	16.0
P3	齒輪	綠	27.0
P4	板手	藍	63.0
P5	撬子	棕	80.0

[NOT] EXISTS的用法

- **EXISTS:**
  - 此運算子在當子查詢有查詢結果產生時，會回傳**TRUE**給主查詢，否則便回傳**FALSE**。
  - 同時，子查詢若有資料產生，會**暫存於記憶體**以待主查詢使用。
- **NOT EXISTS:**
  - EXISTS的反義詞。

- [NOT] EXISTS在標準子查詢中：
  - 主查詢使用的表格與子查詢的表格是完全無關的 (*互*為獨立)。
  - 子查詢是否有結果產生，對主查詢來說只 “有/沒有”、“TRUE/FALSE”，通常只是做為主查詢工作的開關。
    - 若子查詢的回傳結果為TRUE，則主查詢中的SELECT…FROM…便可被執行；反之，便不會被執行。
- [NOT] EXISTS在關聯子查詢中：
  - 主查詢使用的表格與子查詢是有關的。
  - 子查詢是否有結果產生，對主查詢來說除了是 “TRUE/FALSE”的開關功能外，子查詢的結果也會影響主查詢。

※範例題組7※

- [NOT] EXISTS在標準子查詢：

```
SELECT 供應商名稱
FROM Supplier
WHERE EXISTS (SELECT *
              FROM Supplier
              WHERE 城市='台南');
```

供應商(Supplier)

供應商代號	供應商名稱	城市
S1	大勝	台南
S2	冠軍	高雄
S3	無敵	台中
S4	一級棒	高雄

  - 主查詢與子查詢雖用同樣的表格，但彼此無關
  - 子查詢在此僅告知主查詢它是否有回傳值，所以用SELECT *即可 (要用實際欄位名亦可，只是結果相同，沒啥意義)。

供應商(Supplier)

供應商代號	供應商名稱	城市
S1	大勝	台南
S2	冠軍	高雄
S3	無敵	台中
S4	一級棒	高雄

專案供應零件(Project_supp_Component)

供應商代號	零件代號	專案代號	數量
S1	P1	J2	300
S1	P1	J3	100
S1	P4	J1	500
S2	P2	J1	400
S2	P3	J3	600
S3	P1	J2	300
S4	P3	J1	200
S4	P4	J1	700
S4	P5	J2	100

- **[NOT] EXISTS在關聯子查詢：**
  - **依照slide 37的四個表格，列出有供應零件'P1'的供應商名稱。**

Sol:

```
SELECT S.供應商名稱
FROM Supplier AS S
WHERE EXISTS (SELECT *
                FROM Project_supp_Component AS PSC
                WHERE PSC.供應商代號=S.供應商代號 AND
                      PSC.零件代號='P1');
```

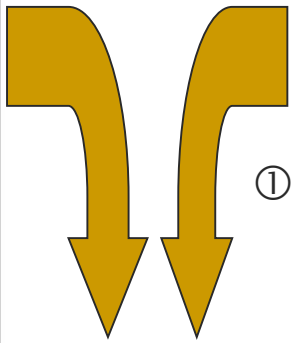
  - **主查詢的表格，在子查詢中會使用到，它將與子查詢之表格做合併處理，所以是有關的。**
  - **子查詢在此除告知主查詢它是否有回傳值，其查詢結果也會影響主查詢。**

專案供應零件(Project_supp_Component)

供應商代號	零件代號	專案代號	數量
S1	P1	J2	300
S1	P1	J3	100
S1	P4	J1	500
S2	P2	J1	400
S2	P3	J3	600
S3	P1	J2	300
S4	P3	J1	200
S4	P4	J1	700
S4	P5	J2	100

供應商(Supplier)

供應商代號	供應商名稱	城市
S1	大勝	台南
S2	冠軍	高雄
S3	無敵	台中
S4	一級棒	高雄



① 子查詢結果

③ 主查詢結果

② 回傳TRUE時，主查詢對象

其它運算子

- **IS NULL**：判斷屬性值是否為NULL
- **BETWEEN…AND…**：指定屬性值必須介於一個最小值(含)與最大值(含)之間
- **LIKE**：利用萬用字元(%及_)做相似字串的比對
  - 百分符號%：表示0~多個任意字元
  - 底線符號_：表示單一個任意字元
  - 例：%大_ _⇨在 “大” 字前可有任意個字元；在 “大” 字後必須恰有兩個字元，比兩個字元多或少都不行!!

※範例題組8※

- 依照slide 37的四個表格，用SQL回答下列問題：
  - 找出所在城市未知的供應商代號與名稱

    Sol:

    ```
    SELECT 供應商代號,供應商名稱
    FROM Supplier
    WHERE 城市 IS NULL;
    ```

  - 找出零件名稱第一個字為 “螺” 的零件之名稱、顏色、重量

    Sol:

    ```
    SELECT 零件名稱, 顏色, 重量
    FROM Component
    WHERE 零件名稱 LIKE '螺%';
    ```

供應商(Supplier)

供應商代號	供應商名稱	城市
S1	大勝	台南
S2	冠軍	高雄
S3	無敵	台中
S4	一級棒	高雄

零件(Component)

零件代號	零件名稱	顏色	重量
P1	螺絲釘	黑	14.0
P2	螺帽	黑	16.0
P3	齒輪	綠	27.0
P4	板手	藍	63.0
P5	撬子	棕	80.0

- 找出專案名稱第二個字為 “星” 的專案之代號、名稱、所在城市

  Sol:

  ```
  SELECT 專案代號, 專案名稱, 城市
  FROM Project
  WHERE 專案名稱 LIKE '_星';
  ```

專案(Project)

專案代號	專案名稱	城市
J1	火星	台中
J2	土星	高雄
J3	太陽	台北

- 找出重量介於16~63的零件之名稱、顏色、重量

  Sol:

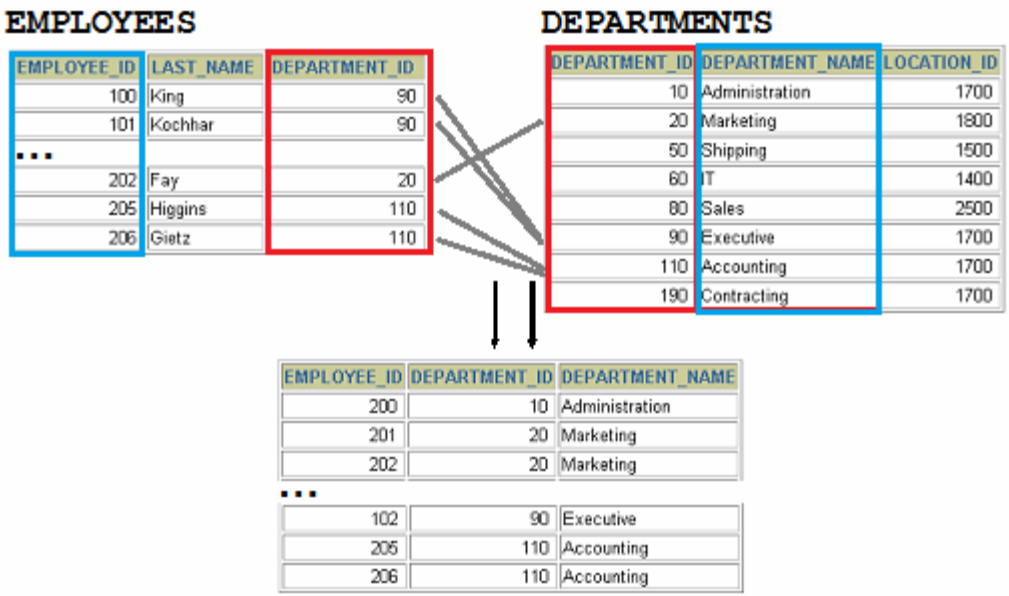
  ```
  SELECT 零件名稱, 顏色, 重量
  FROM Component
  WHERE 重量 BETWEEN 16 AND 63;
  ```

零件(Component)

零件代號	零件名稱	顏色	重量
P1	螺絲釘	黑	14.0
P2	螺帽	黑	16.0
P3	齒輪	綠	27.0
P4	板手	藍	63.0
P5	撬子	棕	80.0

合併查詢(Join)

- 若使用者要查詢的資料來源，是一個以上的表格資料合併在一起的結果，則此一查詢任務需採用合併查詢。



- **合併查詢可分成：**
  - **Inner Join (內部合併)**
    - **將多個表格的內容以某合併條件結合在一起，而不符合條件的資料不予呈現。**
  - **Outer Join (外部合併)**
    - **將多個表格的內容以某合併條件結合在一起，而不符合條件的資料將視需求以不同方式予以呈現。**

Inner Join(內部合併)

- Inner Join常見的寫法，是將結合條件寫在Where子句中。

```
SELECT table01.column01, table02.column02, …
FROM table01, table02
WHERE 結合條件
```

- Inner Join另一種常見的寫法：

```
SELECT table01.column01, table02.column02, …
FROM table01 [INNER] JOIN table02
ON 結合條件
```

※範例題組9※

- 依照slide 37的四個表格，用SQL回答下列問題：
  - 列出位於相同城市的專案名稱、供應商名稱與兩者之城市名稱。
    (學習重點：1. 內部合併; 2. 對表格取別名)

Sol:

寫法 1-

```
SELECT P.專案名稱, P.城市, S.城市, S.供應商名稱
FROM Supplier as S, Project as P
WHERE P.城市= S.城市;
```

專案(Project)

專案代號	專案名稱	城市
J1	火星	台中
J2	土星	高雄
J3	太陽	台北

供應商(Supplier)

供應商代號	供應商名稱	城市
S1	大勝	台南
S2	冠軍	高雄
S3	無敵	台中
S4	一級棒	高雄

寫法 2-

```
SELECT P.專案名稱, P.城市, S.城市, S.供應商名稱
FROM Supplier as S INNER JOIN Project as P
ON P.城市= S.城市;
```

Outer Join(外部合併)

- Inner Join所呈現的查詢結果，為**完全滿足合併條件**。
- 但是，若**不滿足合併條件的資料**也想要查看，那該怎麼辦？
  - 列出位於相同城市的專案資料與供應商資料，但是不符條件的**專案資料**也請一併列出。
  - 列出位於相同城市的專案資料與供應商資料，但是不符條件的**供應商資料**也請一併列出。
  - 列出位於相同城市的專案資料與供應商資料，但是不符條件的**專案資料**與**供應商資料**也請一併列出。

專案(Project)

專案代號	專案名稱	城市
J1	火星	台中
J2	土星	高雄
J3	太陽	台北

供應商(Supplier)

供應商代號	供應商名稱	城市
S1	大勝	台南
S2	冠軍	高雄
S3	無敵	台中
S4	一級棒	高雄

- 外部合併的類型可分為：
  - 左外部合併 (Left Outer Join)
  - 右外部合併 (Right Outer Join)
  - 全外部合併 (Full Outer Join)

專案(Project)

專案代號	專案名稱	城市
J1	火星	台中
J2	土星	高雄
J3	太陽	台北

供應商(Supplier)

供應商代號	供應商名稱	城市
S1	大勝	台南
S2	冠軍	高雄
S3	無敵	台中
S4	一級棒	高雄

```
SELECT table01.column01, table02.column02, …
FROM table01 [LEFT | RIGHT | FULL] [OUTER] JOIN table02
ON 結合條件
```

- MySQL不支援全外部合併指令，但可以用其它方式達到相同的效果。
- 本單元主要講解外部合併之SQL指令操作；外部合併之關聯式代數運作概念及符號表示將於Course 7提到。

※範例題組10※

- 依照slide 37的四個表格，用SQL回答下列問題：
  - 列出位於相同城市的專案資料與供應商資料，但是不符條件的專案資料也請一併列出。

    (學習重點：1. 左外部合併; 2. 對表格取別名)

Sol:

```
SELECT P.專案名稱, P.城市, S.城市, S.供應商名稱
FROM Project as P LEFT OUTER JOIN Supplier as S
ON P.城市= S.城市;
```

專案(Project)

專案代號	專案名稱	城市
J1	火星	台中
J2	土星	高雄
J3	太陽	台北

供應商(Supplier)

供應商代號	供應商名稱	城市
S1	大勝	台南
S2	冠軍	高雄
S3	無敵	台中
S4	一級棒	高雄

- 列出位於相同城市的專案資料與供應商資料，但是不符條件的供應商資料也請一併列出。

  (學習重點：1. 右外部合併; 2. 對表格取別名)

Sol:

```
SELECT P.專案名稱, P.城市, S.城市, S.供應商名稱
FROM Project as P RIGHT OUTER JOIN Supplier as S
ON P.城市= S.城市;
```

專案(Project)

專案代號	專案名稱	城市
J1	火星	台中
J2	土星	高雄
J3	太陽	台北

供應商(Supplier)

供應商代號	供應商名稱	城市
S1	大勝	台南
S2	冠軍	高雄
S3	無敵	台中
S4	一級棒	高雄

- 列出位於相同城市的專案資料與供應商資料，但是不符條件的專案資料、供應商資料也請一併列出。

  (學習重點：1. 全外部合併; 2. 對表格取別名)

Sol:

一般寫法：

```
SELECT P.專案名稱, P.城市, S.城市, S.供應商名稱
FROM Project as P FULL OUTER JOIN Supplier as S
ON P.城市= S.城市;
```

專案(Project)

專案代號	專案名稱	城市
J1	火星	台中
J2	土星	高雄
J3	太陽	台北

供應商(Supplier)

供應商代號	供應商名稱	城市
S1	大勝	台南
S2	冠軍	高雄
S3	無敵	台中
S4	一級棒	高雄

- 不幸的，MySQL沒有支援上述Full Join的指令寫法，但是可用聯集(Union)的做法來模擬，也就是將左、右合併的查詢結果聯集在一起：

```
SELECT P.專案名稱, P.城市, S.城市, S.供應商名稱
FROM Project as P LEFT OUTER JOIN Supplier as S
ON P.城市= S.城市
UNION
SELECT P.專案名稱, P.城市, S.城市, S.供應商名稱
FROM Project as P RIGHT OUTER JOIN Supplier as S
ON P.城市= S.城市;
```

專案(Project)

專案代號	專案名稱	城市
J1	火星	台中
J2	土星	高雄
J3	太陽	台北

供應商(Supplier)

供應商代號	供應商名稱	城市
S1	大勝	台南
S2	冠軍	高雄
S3	無敵	台中
S4	一級棒	高雄

Data Definition Language, DDL (資料定義語言) - (2)

- DDL主要有CREATE, DROP, ALTER三個指令，並可針對以下三個物件進行操作：
  - 資料庫 (database)
  - 表格 (Table)
  - 景觀 (View)
- 景觀 (View):
  - 是由其它表格所衍生出來的關聯表格，View不需要以實體的形式存在，即：View並不需要實際儲存資料，可視為虛擬表格 (Virtual Table)。
  - 其內容是以SELECT指令的執行結果構成，呈現的方式仍是以二維表格為主。所以其定義方式正是以SELECT為基礎

- EMPLOYEES (Base Table)

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
100	Steven	King	SKING	515.123.4567	17-JUN-87	AD_PRES	24000
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89	AD_VP	17000
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	17000
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-90	IT_PROG	9000
104	Br					ROG	6000
1						ROG	4200
						ACCOUNT	6900
						MAN	5800
						CLERK	3500
						CLERK	3100
						CLERK	2600
						CLERK	2500
						SA_MAN	10500
					6	SA_REP	11000
					AR-98	SA_REP	8600
					4-MAY-99	SA_REP	7000
					17-SEP-87	AD_ASST	4400
					17-FEB-96	MK_MAN	13000
				6666	17-AUG-97	MK_REP	6000
205	Shelley	Higgins	SHIGGINS	515.123.8080	07-JUN-94	AC_MGR	12000
206	William	Gietz	WGIETZ	515.123.8181	07-JUN-94	AC_ACCOUNT	8300

View

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY
100	Steven	King	24000
101	Neena	Kochhar	17000
102	Lex	De Haan	17000
103	Alexander	Hunold	9000
104	Bruce	Ernst	6000

建立、查詢與刪除景觀 (View)

- CREATE VIEW: 建立一個新的景觀
  - CREATE VIEW <view name> AS

    SELECT…

    FROM…

    WHERE…
  - 例：建立“列出重量大於螺帽之其它零件的所有資料”的View

```
CREATE VIEW Com_W AS
        SELECT *
        FROM Component
        WHERE 重量 > (SELECT 重量
                      FROM Component
                      WHERE 零件名稱='螺帽');
```

零件(Component)

零件代號	零件名稱	顏色	重量
P1	螺絲釘	黑	14.0
P2	螺帽	黑	16.0
P3	齒輪	綠	27.0
P4	板手	藍	63.0
P5	攝子	棕	80.0

- DROP VIEW: 刪除一個景觀

  DROP VIEW <view name> [CASCADE/RESTRICT]

  - 例：刪除“列出重量大於螺帽之其它零件的所有資料” 的View

    DROP VIEW Com_W;

- 景觀的優點：
  - 隱藏不需要或具私密性的資料
  - 同一關聯可建立多種不同的觀點，讓使用者以不同的角度看同一份資料。
- 景觀的缺點：
  - 景觀的DML操作有諸多限制，無法提供與實際表格完全相同的操作 (畢竟它是屬於虛擬表格)。

Data Control Language, DCL (資料控制語言)

- 此語言主要從事**資料庫的權限控管**，包含Grant, Revoke, Alter Password…等指令。
- **Grant**：建立新的user，並增加該user在資料庫的相關操作權限。

  GRANT <authority> ON <object> TO <users> [identified by "密碼"]
  - 例：
    - GRANT Select
      ON EMPLOYEE(FName,LName)
      TO Jacy1 identified by '123';
    - GRANT Delete
      ON EMPLOYEE
      TO Jacy2;
    - 若沒有設密碼，表示此使用者帳號不需密碼即可登入資料庫系統。

使用者層級權限

權限	意義
CREATE	可建立表格
CREATE TEMPORARY TABLES	可建立暫時性表格
DELETE	可刪除資料列
EXECUTE	可執行程序
INDEX	可建立索引值
INSERT	可加入資料列
LOCK TABLE	可鎖定資料列
SELECT	可選擇資料列
SHOW DATABASES	可執行 SHOW DATABASES 指令
UPDATE	可更新資料列
USAGE	可登入，不過不能做其他動作

管理者層級權限

權限	意義
ALL	除了 WITH GRANT OPTION 之外的所有權限
ALTER	可調整表格
DROP	可刪除表格
FILE	可自檔案中讀入資料
PROCESS	可看見 MySQL 正在執行的程序
RELOAD	可使用 FLUSH 敘述式
REPLICATION CLIENT	可檢查主從的機器在哪
REPLICATION SLAVE	在從屬機器上給予特殊權限進行備份
SHUTDOWN	可執行 mysqladmin shutdown 指令
SUPER	在超過 MySQL 最大連線數的當下仍可登入
WITH GRANT OPTION	賦予其他使用者權限

- **Revoke: 取消某user之權限。**

  REVOKE <authority> ON <object> FROM <users>

  - **例：將使用者Jacy2，對表格employee的delete權限取消掉。**

    ```
    REVOKE Delete
    ON employee
    FROM Jacy2;
    ```

- REVOKE敘述只移除使用者的資料操作權限而非移除使用者，該使用者資訊仍存在於資料庫系統中。
- 要完整的移除使用者，必須使用DELETE指令明確地由資料庫系統相關表格中，將使用者紀錄刪除。

相關指令操作已於MySQL實作課程Chapter3講授示範，請見課程錄影