

CHAPTER 13 資料庫



13-1 資料庫管理系統簡介

13-2 關聯式資料模式和查詢

語言

13-3 Access簡介

13-4 資料探勘

13-5 XML簡介



13-1 資料庫管理系統簡介

- ➡ 資料管理已經是目前各個企業和機構最重要的課題之一。
 - ▶ 舉例來說，銀行必須記錄每個客戶的存款金額及提款紀錄；航空公司必須管理每架班機的飛行時刻表與乘客訂位紀錄；學校則必須記錄學生的學籍資料和選課成績等。
- ➡ 由於資料量龐大，透過電話傳真等人工書面處理，不僅耗時費力，更容易發生人為疏失。所以，將資料數位化並輔以電腦處理，已經是時勢所趨。





13-1 資料庫管理系統簡介

- ➡ 建立數位化的資料處理系統，雖然可利用一般程式語言提供的檔案管理功能，但是當資料量與日俱增之後，就會面臨到下列問題：

資料的重複與
不一致

資料難以存取

資料的限制難以修改





資料的重複與不一致

- ➡ 以學校來說，教務處需要記錄學生的地址以寄發成績單，學務處也需要記錄學生的地址，以寄發兵役通知或其他需要通知監護人的訊息。
- ➡ 若是兩個單位各自開發自己的應用系統，也各自建立檔案維護學生地址資料，則該資料在學校裡被重複儲存。當學生搬家時，就有可能只改了一個單位的資料，而忘了或不知道也需要到另一個單位改資料，造成學校內有兩份不一樣的地址資料，難以判斷何者為真。





資料難以存取

- ➡ 隨著資訊科技的進步，應用系統常會使用不同的程式語言來開發。早期常用的是COBOL，後來則是C語言，但是近年來JAVA語言也很受到歡迎。這些程式語言不僅語法不同，檔案的格式與建立方式也不同。
- ➡ 假設學校10年前開發了一套人事系統，現在希望利用原來的架構，繼續開發新的薪資計算系統，但卻可能發現不知如何使用新的程式語言去讀取舊的檔案格式資料，而造成開發上的困難。





資料的限制難以修改

- ➡ 一般程式語言所提供的檔案功能，都只允許程式設計師描述檔案內每筆**紀錄**(record)由哪些資料格式所組成，及資料大小。
- ➡ 假設課務組在輸入老師的授課時數時，想限定至少要輸入9小時，否則顯示錯誤或不允許輸入，則此功能必須寫死在程式碼裡。若5年後，老師的授課時數下限由9小時改成6小時，則程式設計師必須從眾多程式碼中，找出對應的限制式，把「9」改成「6」，這是一件很辛苦的工作。



資料的限制難以修改

- ➡ 要是當時使用的程式語言已經過時，則帶來的問題更大。
- ➡ 這些問題的產生，是因為一般程式語言是所謂的「功能」導向，重點在於寫出正確且結構化的「程式碼」，達到使用者所希望的功能，但是卻缺乏對整個系統所使用「資料」的分析工具。





13-1 資料庫管理系統簡介

- ➡ 當資料日漸複雜，使用者越來越多，在系統方面也會面臨到很多問題。
- ➡ 以下我們提出一些常見的問題，並討論資料庫系統的作法：

資料異動的
一致性

併行存取資
料的錯誤

安全控管的
困難





資料異動(transaction)的一致性

- ▶ 當利用ATM做跨行轉帳時，雖然就使用者看來，是一個簡單的動作，但是影響到的卻是來自兩個銀行的兩個帳戶。
- ▶ 假設從A銀行的帳戶轉帳1萬元到B銀行的帳戶，裡面其實包含兩個動作：
 - ▶ 從A銀行的帳戶扣款1萬元。
 - ▶ 將B銀行的帳戶增加1萬元。





資料異動(transaction)的一致性

- ➡ 若做完第一個扣款動作之後，系統因為停電或其他原因突然故障，而沒有繼續執行第2個動作，等到系統恢復正常執行後，若不對這兩個帳戶給予特別的處理，則使用者一定會加以抗議，因為他的總錢數變少了，也就是資料庫的「一致性」不再存在。
- ➡ 一般資料庫系統的作法會復原(recover)第1個動作的結果，也就是把1萬元再還給A銀行的帳戶。





併行存取資料的錯誤

- ➡ 大型資料庫通常會有很多使用者同時存取，如同航空公司的訂位系統，一般可透過很多個旅行社進行交易。
 - ▶ 假設甲先生從台北的旅行社，要求在1月1日台北到舊金山的航次訂位；而乙小姐透過高雄的旅行社，要求在同一航班上訂位。
 - ▶ 如果兩人訂位時，第10排座位A是空位，若是不加以控管，則可能兩人會正好都訂到該位置。
- ➡ 一般資料庫系統的作法，是利用鎖定(lock)的機制，允許大家可以同時讀取資料，但是碰到寫入的動作時，同一時間則允許只有一個人進行。





安全控管的困難

- ➡ 資料庫系統裡常常整合來自不同單位的資料，譬如學校的系統可能整合人事薪資資料，以及學生的成績資料。
- ➡ 在目前資訊安全很重要的時代，我們希望限定各個單位的人，只看到其負責的部分，譬如限定人事室的職員只看到人事資料，會計室只看到薪資資料，教務處只看到成績資料。





安全控管的困難

- ➡ 這方面的控管若是都利用程式碼來限制，既困難又不容易修改。
- ➡ 但是在目前的資料庫系統，都可直接指定每個使用者或每個群組所能看到的資料，在使用權限的設定和修改方面都相當方便。





13-1 資料庫管理系統簡介

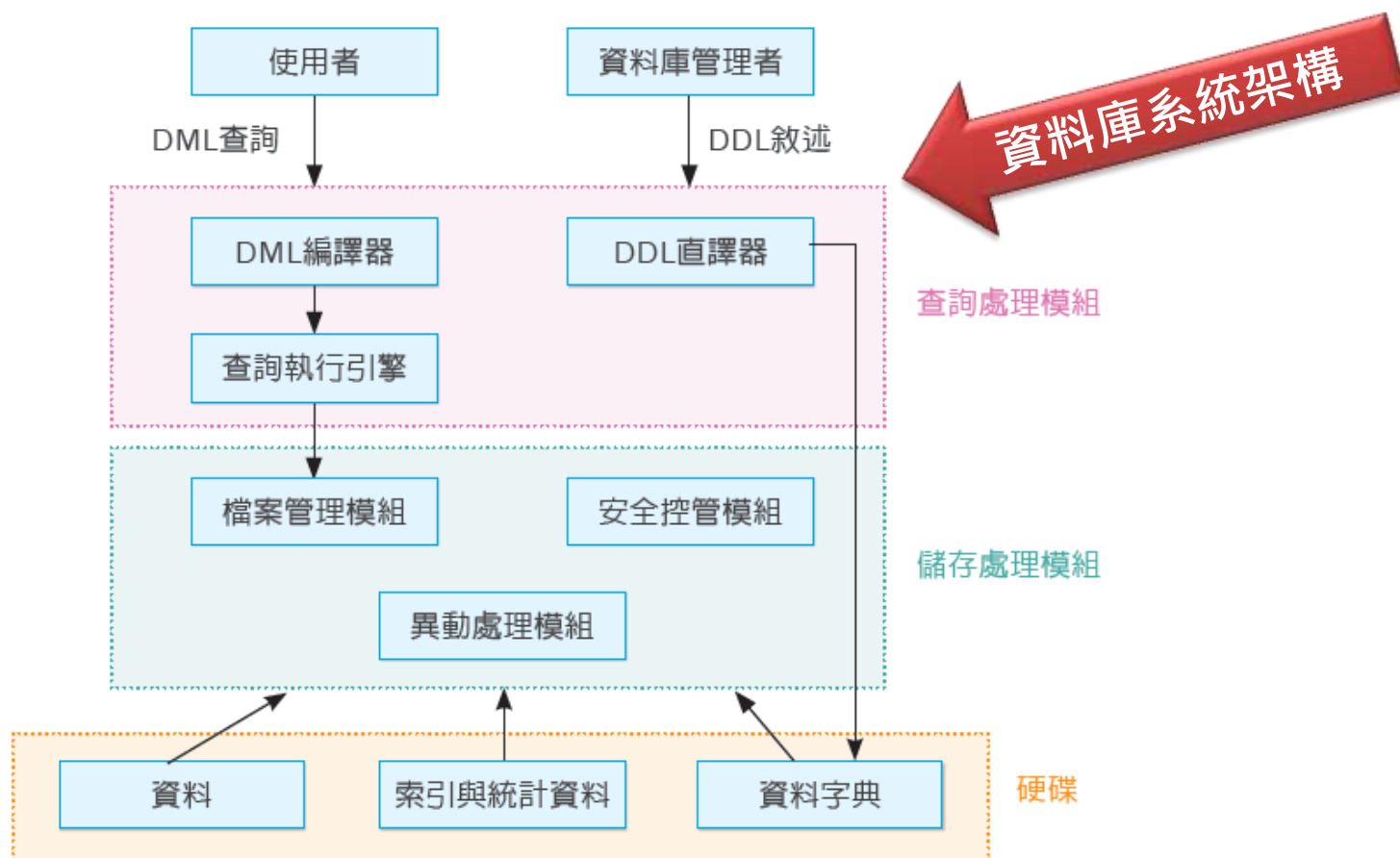
- ▶ 一般資料庫管理系統的架構，其中包含幾個成分：

查詢處理模組

儲存處理模組



13-1 資料庫管理系統簡介





查詢處理模組(query processor)

- ▶ 所謂的「查詢語言」，可看作是使用者和資料庫系統之間的溝通橋樑，其中又可細分為：

資料定義語言(Data Definition Language ; DDL)

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





查詢處理模組(query processor)

- ➡ 目前通用的查詢語言為SQL(Structured Query Language)。
- ➡ 「查詢處理模組」的功用，主要是接受使用者下達的查詢句，利用「編譯器」將其解析之後，透過「查詢執行引擎」選擇最有效率的執行方式，再交給「儲存處理模組」將資料取出。





儲存處理模組(storage manager)

- ➡ 由於資料庫的資料是以檔案的方式存放在硬碟中，所以此模組會先呼叫「檔案管理模組」，找出資料存放的檔案。
- ➡ 同時在執行的過程中，「安全控管模組」會檢查使用者的權限，避免惡意的資料破壞；而「異動處理模組」會確定整個資料庫內容的一致性和正確性。





儲存處理模組(storage manager)

- ➡ 圖13-1除了標示出一般的使用者外，還特別標示出**資料庫管理者** (DataBase Administrator ; DBA)。
- ➡ 我們可以看到一般使用者是下達DML查詢句，以便對資料做查詢等處理；而資料庫管理者，則是下達DDL敘述句，來定義資料庫內建立的資料型態和關係。之所以需要特別的資料庫管理者，是因為資料庫系統相當龐大，所以需要具有專門技術且瞭解整個系統的人，來負責管理。





儲存處理模組(storage manager)

- ➡ 資料庫管理者的職責至少包含了下列幾項：
 - ▶ 決定哪些資料包含在資料庫內，且設定資料彼此之間的關聯。
 - ▶ 設計資料存放在硬碟裡面的架構。
 - ▶ 建立使用者帳號。
 - ▶ 執行安全的控管。
 - ▶ 週期性的資料維護，譬如：將資料備份、確定硬碟空間是否足夠、監控系統的效能並做適當的調整。





13-2 關聯式資料模式和查詢語言

- ➡ 為了將真實世界的資料以資料庫軟體表示，我們需要適當的資料表示工具，稱作**資料模式**(data model)。
- ➡ 最早被提出來的為**網路模式**(network data model) 和 **階層模式**(hierarchical data model)，但是最具影響力的卻是1970年由Codd提出的**關聯式模式**(relational data model)。





13-2 關聯式資料模式和查詢語言

- ➡ 關聯式資料庫主要是由很多關聯(relation)所組成，一個關聯就如同一個表格，由「列」和「欄」所構成，在本章中我們會將「關聯」和「表格」這兩個用詞視為相同。





13-2 關聯式資料模式和查詢語言

- ▶ 一個學生(student)關聯如下表所示，其中第一列代表了這個關聯所要表示的資料特性，稱作屬性(attribute)，而每一個屬性各自對應到一欄。
- ▶ 在此關聯中，每個學生我們希望記錄他的系別、年級、學號、姓名、地址、監護人、成績排名等屬性。
- ▶ 其餘的每一列則代表了這個關聯集合裡的某一筆資料，稱作資料列(tuple)，我們在每一列的前面加註編號以方便日後的說明。





13-2 關聯式資料模式和查詢語言

- ▶ 舉例來說，編號1的那一列，表示了某個特定學生，其系別為「資工系」、年級為「4」、學號為「B9901」、姓名為「王雅蕙」、地址為「台北市」、監護人為「王爸爸」、排名為「1」等等；至於編號2的那一列，則表示了另一個姓名為「劉維新」的學生的相關資料。





13-2 關聯式資料模式和查詢語言

	系別	年級	學號	姓名	地址	監護人	排名
1	資工系	4	B9901	王雅蕙	台北市	王爸爸	1
2	資工系	4	B9902	劉維新	台中市	劉大新	11
3	資工系	4	B9903	張自強	高雄市	張善良	21
4	電機系	4	B9904	施小龍	台北市	施大龍	7
5	電機系	4	B9905	林正當	台中市	林正正	2
6	電機系	4	B9906	鄭順利	高雄市	鄭大順	15
7	資工系	4	B9907	林紹興	台北市	林爸爸	13
8	資工系	4	B9908	洪志堅	台北市	洪媽媽	6
9	資工系	4	B9909	陳柏豪	台北市	陳阿姨	30
10	資工系	4	B9910	張建設	高雄市	張成功	4

學生(student)關聯





13-2 關聯式資料模式和查詢語言

- ➡ 將資料建立好之後，我們必須透過特殊的語言將資料查詢出來，此類語言稱作**查詢語言**(query language)，而標準的關聯式查詢語言稱作SQL(Structured Query Language)，以下簡介其語法。
- ➡ 一個SQL查詢句主要是由三個部分所構成，分別稱作SELECT子句、FROM子句和WHERE子句。





13-2 關聯式資料模式和查詢語言

- ➡ SELECT子句列舉欲顯示給使用者的屬性、所參考到的關聯表示在FROM子句、而資料列的選擇條件則寫在WHERE子句。
- ➡ 換句話說，根據FROM子句，找出會使用到的關聯；再根據WHERE子句的條件式，挑出該關聯裡符合限制的資料列；最後依據SELECT子句，將這些資料列的特定屬性輸出給使用者。





13-2 關聯式資料模式和查詢語言

- 以表13-1 的學生關聯為例，假設要輸出學號「B9901」同學的地址與監護人，則所對應的SQL查詢句如下所示：

查詢句1

```
SELECT 地址, 監護人  
FROM student  
WHERE 學號 = 'B9901'
```





13-2 關聯式資料模式和查詢語言

- 在查詢句1當中，將要使用到的關聯表格student列在FROM子句；至於WHERE子句的限制式，則是為了限定學號，所以根據表格13-1，編號1的資料列會被挑選出來；接著根據SELECT子句，將該筆資料列在「地址」和「監護人」屬性欄位的值輸出，所得則為下表。

地址	監護人
台北市	王爸爸

查詢句1的輸出





13-2 關聯式資料模式和查詢語言

- SQL語言允許很多種類的條件式表示在WHERE子句裡，其中可利用不同的算數運算子 (arithmetic operator)，如「>」、「<」等；或使用邏輯運算子 (logical operator)，如「and」、「or」、「not」等。





13-2 關聯式資料模式和查詢語言

- ➡ 查詢句2是選出所有在系上排名前10名的同學學號和姓名
- ➡ 使用「<」這個算數運算子：

```
查詢句2  
SELECT 學號, 姓名  
FROM student  
WHERE 排名<=10
```





13-2 關聯式資料模式和查詢語言

- 由於很多筆資料列在「排名」欄位的值小於10，所以符合條件的共有5筆，也就是表13-1中的第1、4、5、8、10筆資料列。從這些資料列中選出「學號」和「姓名」欄位，則所得的結果如下表所示。

學號	姓名
B9901	王雅蕙
B9904	施小龍
B9905	林正當
B9908	洪志堅
B9910	張建設

查詢句2的輸出





13-2 關聯式資料模式和查詢語言

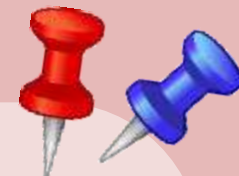
- ▶ 若是使用者只希望針對「資工系」的學生查出排名前10名的同學，則所對應的SQL查詢句在WHERE子句裡，必須利用「and」連接詞，來要求兩個限制條件都成立，所對應的SQL查詢句如下：

查詢句3

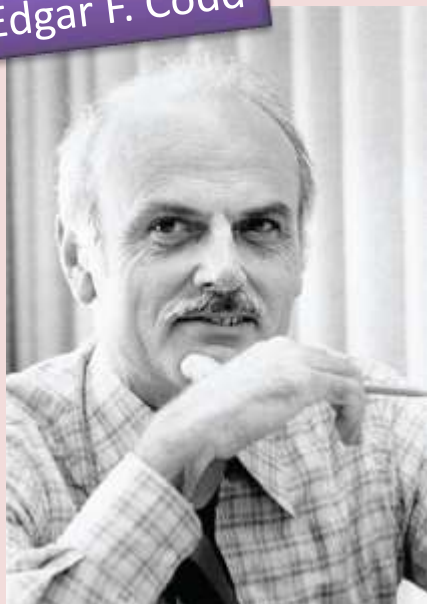
```
SELECT 學號, 姓名  
FROM student  
WHERE 排名<=10 and 系別 = '資工系'
```



IT 專家



Edgar F. Codd



1965年Codd在密西根大學取得博士學位。畢業後Codd加入IBM在矽谷的實驗室，他發現當時的資料庫管理系統，大量仰賴指標將資料串接起來，毫無理論和章法，所以他就基於數學邏輯的理論基礎，於1970年創造了關聯式模式。但是，由於當時IBM內部絕大部分還是支持傳統的資料模式，直到數年後公司才開始重視Codd的想法，而於1981年設計出相關的查詢語言SQL，並於1983年實作出關聯式資料庫系統DB2。至於Larry Ellison創辦的Oracle公司，則是依據關聯式模式建造出第一個商業用的資料庫軟體系統，而成為一個相當成功且賺錢的公司。





13-2 關聯式資料模式和查詢語言

- ➡ 此查詢句的輸出結果則會只包含3筆資料列，分別對應到學號「B9901」、「B9908」、「B9910」的同學。
- ➡ 以上討論的情況是只有針對一個關聯，但是資料庫系統裡通常會建立很多不同的表格，管理不同類型的資料。





13-2 關聯式資料模式和查詢語言

- 下表中的成績(enroll)關聯，是記錄學生修課的成績。

學號	課程	作業	期中考	期末考	總成績
B9901	資料庫	65	90	73	80
B9901	程式語言	0	33	49	40
B9902	程式語言	84	48	36	70
B9904	資料庫	71	51	38	60
B9904	程式語言	53	68	78	71
B9905	作業系統	59	41	79	65

成績(enroll)關聯





13-2 關聯式資料模式和查詢語言

- ➡ 在成績關聯裡，每位同學是以其學號作代表，如果想要以姓名為基準，知道每個同學修習了哪幾門課，就必須同時參考到學生(student)關聯和成績(enroll)關聯。
- ➡ 一個常見的錯誤，是直接把兩個關聯寫在FROM子句裡，而不加以限制，如同下面這個查詢句

```
SELECT *  
FROM student, enroll
```





13-2 關聯式資料模式和查詢語言

- ➡ 這裡我們在SELECT子句裡使用「*」這個符號，是希望輸出這些資料列的所有屬性
- ➡ 這個查詢句所產生的輸出，是學生關聯的10筆資料列，和成績關聯裡的6筆資料列所有可能的組合，也就是會產生出60(10×6)筆資料列。





13-2 關聯式資料模式和查詢語言

- ➡ 下表中只列舉60筆中的12筆作為範例，其中有些省略掉的欄位以「...」表示。
- ➡ 這裡可以觀察到許多不合理或無意義的資料列
 - ▶ 譬如，前6筆資料列都代表了「王雅蕙」同學，但是在第3筆到第6筆資料列中，卻發現該同學在「student.學號」的欄位值，和「enroll.學號」的欄位值，並不相同，也就是一個同學出現兩個不同的學號。
 - ▶ 同樣的情況，針對「劉維新」同學，也可在第7、8、10、11、12筆資料列觀察到。





13-2 關聯式資料模式和查詢語言

	...	student.學號	姓名	enroll.學號	課程	...
1	...	B9901	王雅蕙	B9901	資料庫	...
2	...	B9901	王雅蕙	B9901	程式語言	...
3	...	B9901	王雅蕙	B9902	程式語言	...
4	...	B9901	王雅蕙	B9904	資料庫	...
5	...	B9901	王雅蕙	B9904	程式語言	...
6	...	B9901	王雅蕙	B9905	作業系統	...
7	...	B9902	劉維新	B9901	資料庫	...
8	...	B9902	劉維新	B9901	程式語言	...
9	...	B9902	劉維新	B9902	程式語言	...
10	...	B9902	劉維新	B9904	資料庫	...
11	...	B9902	劉維新	B9904	程式語言	...
12	...	B9902	劉維新	B9905	作業系統	...

學生表格和成績表格直接組合的部分結果





13-2 關聯式資料模式和查詢語言

- ➡ 所以，正確的SQL寫法，是將這兩個表格，以適當的屬性串連起來，稱作兩個表格的連結(join)。
- ➡ 以此例而言，學生關聯裡記錄了每個學生的基本資料，包含了「學號」等；而成績關聯裡，每個選課紀錄裡，也是以「學號」來代表修課的學生，所以可以使用兩個表格共同的「學號」屬性，作為連結的基礎。





13-2 關聯式資料模式和查詢語言

- 兩個表格的連結是以WHERE子句裡的一個「相等」限制式來表示，如同下面所列的SQL查詢句：

查詢句4

```
SELECT 姓名, student.學號, 課程  
FROM student, enroll  
WHERE student.學號 = enroll.學號
```



13-2 關聯式資料模式和查詢語言

- 在此查詢句中，針對連結後的結果，只需要「姓名」、「學號」和「課程」三個屬性，則所輸出的資料如下表所示。

姓名	student.學號	課程
王雅蕙	B9901	資料庫
王雅蕙	B9901	程式語言
劉維新	B9902	程式語言
施小龍	B9904	資料庫
施小龍	B9904	程式語言
林正當	B9905	作業系統

查詢句4的輸出





13-2 關聯式資料模式和查詢語言

- ➡ 將兩個表格連結之後，所產生的資料列可能比原先表格的資料列多，也可能比原先表格的資料列少。
- ➡ 以學生表格為例，每位同學只以一筆資料列表示，但是連結後，有的同學，如「王雅蕙」，會產生出兩筆資料列；而另一方面，學號「B9903」的「王自強」同學，由於在成績表格中沒有任何紀錄，所以在上表中並沒有對應的資料列。





13-2 關聯式資料模式和查詢語言

- ➡ 以成績關聯為例，連接之後的資料列個數會正好和原先的資料列個數相同，這是因為成績關聯的每一筆資料列利用學號值到學生關聯尋找對應的資料列時，會正好找到一個學生資料列來組合。
- ➡ 以下再探討一個常見的SQL寫法，也就是先從一個表格挑選特定資料列，再到另一個表格抓取對應的相關資料。





13-2 關聯式資料模式和查詢語言

- 下面這個SQL句子會只取出「王雅蕙」同學所修習的課程，也就是表13-6中的前兩筆資料列。
- 注意到WHERE子句裡，以「and」連結的限制式，前後的順序並沒有關係：

查詢句5

```
SELECT 姓名, 課程  
FROM  student, enroll  
WHERE student.學號=enroll.學號 and 姓名 = '王雅蕙'
```



13-2 關聯式資料模式和查詢語言

- 下面這個SQL句子會取出「王雅蕙」同學修習「資料庫」這門課的成績：

查詢句6

```
SELECT 姓名, 課程, 作業, 期中考, 期末考, 總成績  
FROM student, enroll  
WHERE student.學號=enroll.學號 and  
       姓名 = '王雅蕙' and  
       課程 = '資料庫'
```



13-2 關聯式資料模式和查詢語言

- ➡ 所得到的結果如下表所示：

姓名	課程	作業	期中考	期末考	總成績
王雅蕙	資料庫	65	90	73	80

查詢句6的輸出

- ➡ 在查詢句5和查詢句6中，由於「學號」欄位在student關聯和enroll關聯裡都被定義，所以在該欄位之前，利用符號「.»加註來源表格，以避免產生混淆，其餘的欄位則加註與否皆可。



13-2 關聯式資料模式和查詢語言

- ➡ SQL DDL的語法，它提供了建立資料表綱要(schema)的功能。
- ➡ 在定義一個關聯的綱要時，除了提供此關聯和所有屬性的名稱，每個屬性的資料型態及資料大小，都必須加以指定。

```
create table student (  
    系別      char (6),  
    年級      char (1),  
    學號      char (5),  
    姓名      varchar (10),  
    地址      varchar (20),  
    監護人    varchar (10),  
    排名      integer  
)
```





13-2 關聯式資料模式和查詢語言

- ➡ 學生關聯共包含了7個欄位：系別、年級、學號、姓名、地址、監護人、排名。在關聯式資料模式中，資料型態以字串和數字為主。
- ➡ 數字分為整數和實數等，和一般程式語言提供的型態類似；而字串型態則可分為char和varchar兩種。這兩種字串型態的差別，在於前者會使用所有宣告的空間；而後者則只會使用到輸入資料大小的空間。





13-2 關聯式資料模式和查詢語言

- ➡ 舉例來說，指定「系別」欄位的型態為 `char(6)`，如果你只輸入5個字母的話，則系統會自動補一個空白，讓每個系別的欄位值都正好佔據6位元組的空間。
- ➡ 另外，將「地址」欄位的資料型態定義為 `varchar(20)`，這樣的話，使用者最多可以輸入20個字母，但是若輸入15個字母的話，系統只使用15個位元組的空間。
- ➡ 值得注意的是：一個中文字需使用兩個位元組的空間。





13-2 關聯式資料模式和查詢語言

- 在定義表格時，還可以進一步指定表格內資料的限制，最常見的限制是**主鍵**(primary key)和**外來鍵**(foreign key)的限制。

「主鍵」是定義在某一個表格上，它可以由一個屬性或多個屬性所構成，這個(或這些)屬性能成為主鍵的條件，是在任何情況下，它們的屬性值在整個表格裡都不會重複。

「外來鍵」雖然也是定義在某一個表格上，但它卻表示了和另一個表格之間的「從屬」關係。





13-2 關聯式資料模式和查詢語言

➡ 定義了主鍵和外來鍵的學生表格和成績表格如右圖。

```
create table student (  
    系別      char (6),  
    年級      char (1),  
    學號      char (5),  
    姓名      varchar (10),  
    地址      varchar (20),  
    監護人    varchar (10),  
    排名      integer),  
    primary key (學號)  
)  
  
create table enroll (  
    學號      char (5),  
    課程      varchar (10),  
    作業      integer,  
    期中考    integer,  
    期末考    integer,  
    總成績    integer,  
    primary key (學號, 課程),  
    foreign key (學號) references student  
)
```



13-2 關聯式資料模式和查詢語言

- ➡ **主鍵**限制了某些屬性值在同一個表格不可重複，其「唯一」的特性方便我們用來取出一筆特定的資料列，在資料表格定義中是用到 **primary key** 這個關鍵詞。
- ➡ **外來鍵**則是限制了兩個表格建立資料時的先後關係，在資料表格定義中是用到 **foreign key** 和 **references** 這兩個關鍵詞。





資訊科技專欄



隨著資料庫軟體日漸受到歡迎與重視，以作業系統起家的微軟公司，正式於1993年，首度推出在Windows NT上運行的SQL Server。該軟體剛問世時，一般企業還是持保留態度，不確定該資料庫伺服器是否能安全且有效率地處理大量的資料。

但是，如同大家所熟悉的Office和Windows系統，SQL Server提供了易於操作的圖形式介面，再加上微軟公司推出低價策略，所以SQL Server逐漸被市場所接受，日後也不斷地推陳出新。





13-3 Access簡介

- ➡ 建立資料庫
- ➡ 建立資料表
- ➡ 設計檢視
- ➡ 設定主鍵和外來鍵
- ➡ 建立SQL查詢





13-3 Access簡介

- ➡ Access是美國微軟公司所發展的資料庫軟體，相對於微軟另一個資料庫軟體 SQL Server，Access比較適合處理小型的資料庫。
- ➡ 當Access軟體發展到7.0版之後，由於其軟體功能強大且易於學習，同時越來越多企業甚至是個人，都體認到資料管理的重要性，所以日後Access就併入到廣受歡迎的Office系列，以便與其他Office成員如Word、Excel之間的資料整合更為方便。





13-3 Access簡介

- ▶ 另一方面，Access希望讓一般使用者可以快速地開發以資料庫為基礎的應用系統，所以也針對Web應用或者與SQL Server連結方面，提供了更便利且快速的開發環境。





13-3 Access簡介

► 啟動Access的方法



透過【開始】
功能表，即可點選【Access】
將Access啟動。





建立資料庫

- ➡ 進入Access之後，首先必須建立【資料庫】，然後才能建立【資料表】，進而建立【查詢】、【報表】等相關物件。
- ➡ 注意到在Access中的「資料表」等同於上節中提到的「關聯」或「表格」。
- ➡ Access軟體附帶了數個範本資料庫，欲對整個資料庫系統應用有整體瞭解的讀者，可先進入到該範例一窺究竟。





建立資料庫

- ➡ 在Access啟動畫面的中央，可看到在預設的情況下，【空白桌面資料庫】圖示已經被選取。直接雙擊該圖示，則會在預設的目錄中
- ➡ 譬如：「C:\Users\user\Documents」，利用預設的檔名，譬如：「Database1.accdb」，建立一個新的資料庫。





建立資料庫

- ▶ 若要改變儲存的位置或檔名，可以點選【檔案】功能表中的【另存新檔】進行操作，方式和一般Office軟體一致。





建立資料庫

- ➡ 假設我們現在要建立一個「學校」資料庫，所對應的檔名是「School.accdb」。
- ➡ 建立此資料庫之後，以後我們在Access中透過檔名「School.accdb」，就可以載入該資料庫，如同開啟一個已經存在的Word檔案一樣。





建立資料庫

- ➡ Access 2003版本之前所建立的資料庫，副檔名為「mdb」，此類型的檔案在Access 2007之後的版本中仍然被支援。
- ➡ 而「accdb」和「mdb」類型資料庫的差別，在於前者支援更多新的功能，所以為Access 2007之後版本的預設檔案類型。





建立資料庫

- ➡ 若是開啟一個先前建立的資料庫，會進入到【資料庫視窗】。
- ➡ 在該視窗上方的標籤中，將功能表區分成【檔案】、【常用】、【建立】、【外部資料】、【資料庫工具】、【說明】等頁面。
- ➡ 注意，功能表會隨著你所執行的工作不同而隨之改變。

資料庫視窗的功能表





建立資料庫

- ➡ 在【建立】頁面中，提供了建立【範本】、【資料表】、【查詢】、【表單】、【報表】、【巨集與程式碼】等相關功能。





建立資料庫

- ➡ 資料表是資料庫的基本單位。基本上，建立資料表有以下幾種方式：
 - ▶ **【資料表】選項**：位於【建立】頁面內的【資料表】類型中，提供藉由輸入資料以建立資料表的方式。
 - ▶ **【資料表設計】選項**：位於【建立】頁面內的【資料表】類型中，提供使用【設計檢視】視窗建立資料表。
 - ▶ **【SharePoint清單】選項**：位於【建立】頁面內的【資料表】類型中，可連結至 SharePoint Server上所建立的清單或將其匯入。





建立資料庫

- ▶ **【應用程式組件】** 選項：位於【建立】頁面內的【範本】類型中，可利用Access提供的內建範例資料表和表單等元件，將其修改成所要的資料表。
- ▶ **連結或匯入外部資料**：此類功能位於【外部資料】的頁面中，可將外部的資料如Excel檔案等，連結或匯入Access中。





建立資料表

- ➡ 選取【建立】頁面中的【資料表】選項後，會出現一個空白資料表。
- ➡ 同時視窗上方的功能表會多出一個【資料表工具】，內含【欄位】和【表格】兩個頁面，同時看到在此空白的資料表中，其內建的第一個欄位名稱為【識別碼】，其作用為自動產生不會重複的數值，是Access替資料表預設的主索引鍵欄位。





建立資料表

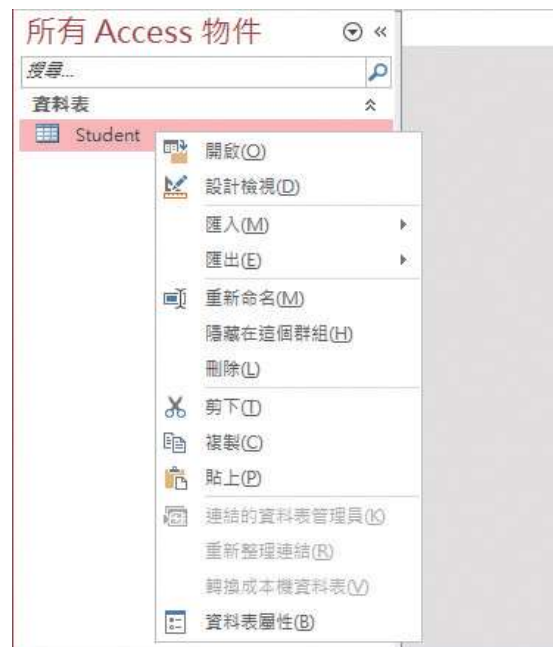
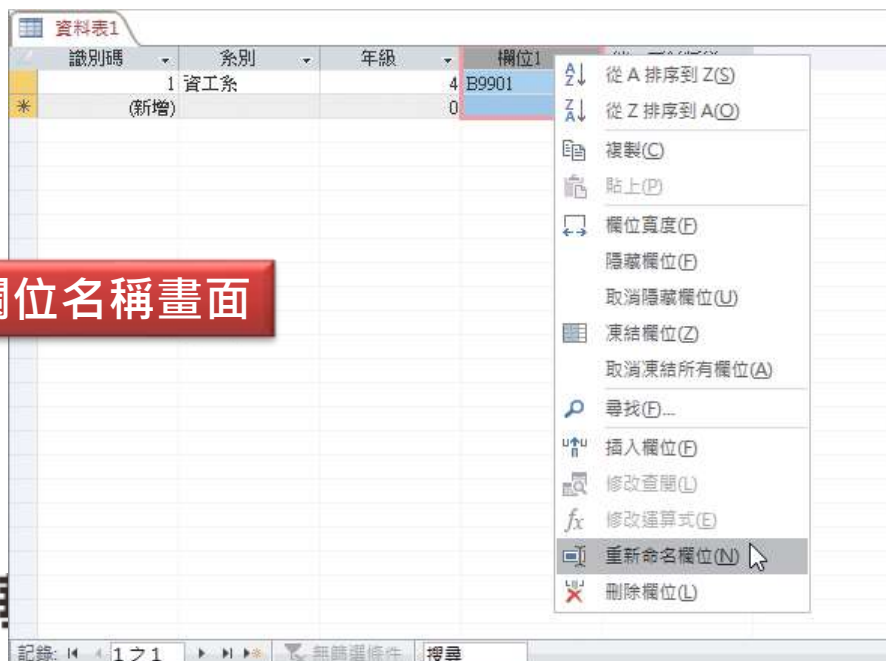
- 一開始指標會停在第二欄，若在該處輸入資料，則其欄位會自動改名為預設名稱【欄位1】，依此類推。
- 也可選取某一個欄位，然後按滑鼠右鍵，選擇【重新命名欄位】，再輸入有意義的欄位名稱。





建立資料表

- 下圖顯示我們已經建立一筆資料列的部分資料，並且將其中兩個欄位名稱更改為「系別」和「年級」，而且正準備更改下一個欄位的名稱。






建立資料表

- ➡ 也可以在選取某個欄位之後，執行插入新的一欄或刪除該欄等等動作。
 - ▶ 注意，在此列資料中，【識別碼】內的數值1是Access所主動產生的。
- ➡ 由於該欄位為資料表預設的主索引鍵，若要將其刪除並利用其他有意義的欄位作為主索引鍵，必須在【設計檢視】中才能進行。



建立資料表

- ▶ 等到輸入完一筆筆的資料列，並確定欄位個數與每個欄位的名稱之後，就可以按一下視窗右上角的 ，這時Access會跳出另一個視窗詢問你是否要儲存在資料表1的設計變更，注意到，「資料表1」是系統先幫你取的暫時名稱，我們在此回答【是】。
- ▶ 接下來，系統會跳出【另存新檔】的對話方塊，我們在此鍵入表格的名稱「Student」之後則可加以儲存。



建立資料表

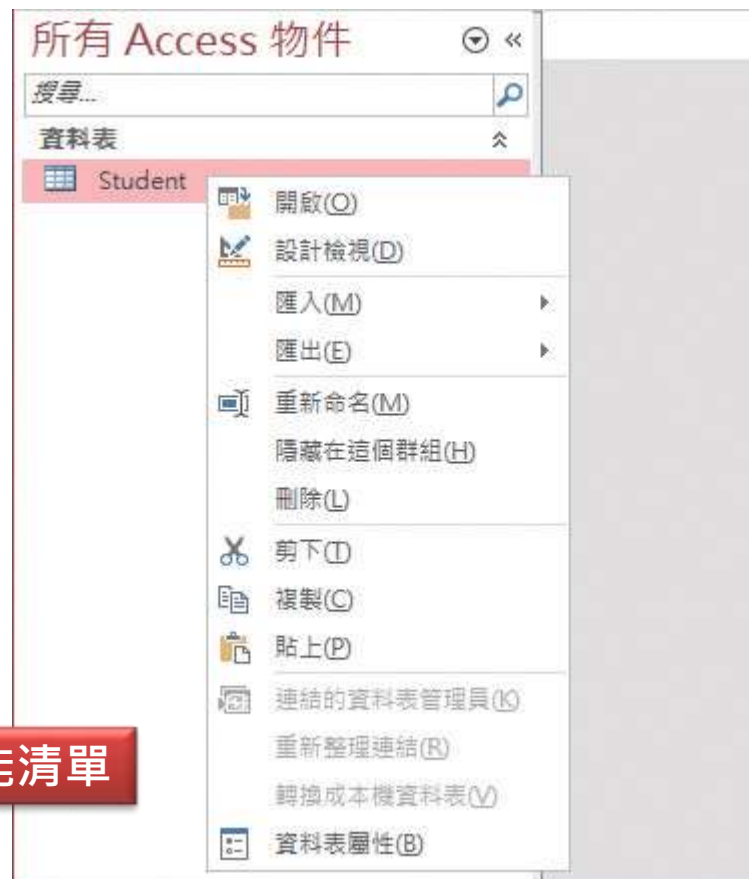
- ➡ 執行完此步驟，在資料庫視窗的左側，就會出現【Student】資料表，我們點選之後按滑鼠右鍵，即可進行對該資料表的操作。
- ➡ 在此若點選【開啟】則可再度顯示此資料表的內容，並進行資料的輸入與維護。
 - ▶ 注意，資料庫視窗左邊的窗格中，會列出目前在此資料庫內已經建立的相關物件。





建立資料表

下圖表示了目前 **School** 資料庫裡面只建立了一個 **Student** 資料表。



可對資料表操作的功能清單





設計檢視

- ➡ 當我們在【建立】頁面中點選【資料表設計】，或者在如上圖的功能清單中點選【設計檢視】，就會進入【設計檢視】視窗。
- ➡ 在此視窗中提供了設計資料表最完整的功能，雖然對一般初學者而言比較困難，卻很值得深入瞭解，所以我們用較多的篇幅來介紹此功能。





設計檢視

➡ 設計表格的順序一般如下：

1. 設計個別欄位：

- ▶ 輸入欄位名稱

- ▶ 選取資料類型：如字串或數值

- ▶ 設定欄位資料的一般限制

2. 設計整個表格內欄位間的限制：如主鍵(primary key)

3. 設計表格間的限制：如外來鍵(foreign key)





設計檢視

- ▶ 假設我們同之前所介紹，利用【資料表】選項提供的功能建立了學生表格，但是之後想針對「排名」欄位，限定裡面的值必須是大於0的整數，則可以在資料庫視窗中，選取【Student】資料表物件後按右鍵，然後再選取【設計檢視】選項，則會進入【設計檢視】畫面。





設計檢視

- ➡ 這裡可以看到在【欄位名稱】的部分列出了Student資料表的8個欄位，而且【資料類型】也分別被指定好。
- ➡ 讀者或許會奇怪，之前我們並沒有替Student資料表的欄位指定資料類型，其實這些類型是Access根據你輸入的資料自動幫你設計的。





設計檢視

- ➡ 在【設計檢視】裡，可以自行選擇或更改成其他資料類型。
- ➡ 以下大致說明幾項Access提供的資料型態：
 - ▶ 自動編號：自動插入的唯一順序值，有時具有主鍵的功能。
 - ▶ 簡短文字：資料為字串，最長可包含255個字元。
 - ▶ 長文字：可輸入一長段文章，最長為63999個字元。





設計檢視

- ▶ **數字**：資料為可計算之數值，可根據資料值的範圍或是否允許小數點，選擇「整數」、「長整數」、「單精準度」或「雙精準度」等。
- ▶ **日期/時間**：其中包含年、月、日、時、分、秒。
- ▶ **是 / 否**：只能在二種值中選一，譬如「是 / 否 (Yes/No)」、「真 / 假 (True/False)」或「開 / 關 (On/Off)」。
- ▶ **OLE 物件**：其他程式建立的 OLE 物件，如圖形或 Word 文件，可連結或內嵌在 Microsoft Access 資料表之中。





Student			x
欄位名稱	資料類型	描述 (選用)	
系別	簡短文字		
年級	數字		
學號	簡短文字		
姓名	簡短文字		
地址	簡短文字		
監護人	簡短文字		
排名	數字		

欄位屬性

一般 查閱

欄位大小	長整數
格式	
小數位數	自動
輸入遮罩	
標題	
預設值	0
驗證規則	
驗證文字	
必須有資料	否
索引	否
文字對齊	一般

設計檢視畫面

欄位名稱最長可達 64 個字元，包含空格。請按 F1 鍵查看欄位名稱的相關說明。





設計檢視

- ➡ 在上圖視窗的左下方，列舉了Access提供的限制欄位值的方式。如同之前的範例，我們希望「排名」的所有欄位值都必須是大於0，所以必須先選取「排名」欄位，接著在【驗證規則】那個格子中，填入一個數學限制式 >0 。
- ➡ 由於表格內已經存在了之前輸入的資料，所以在離開設計檢視畫面時，Access會詢問是否要利用新建立的限制式去檢查既有的資料，通常我們會回答【是】以便檢查資料的正確性。





設計檢視

- ➡ 視窗左下方還有其他的限制方式，以下針對比較常用的加以說明：
 - ▶ 格式
 - ▶ 輸入遮罩
 - ▶ 預設值
 - ▶ 驗證規則
 - ▶ 驗證文字
 - ▶ 必須有資料
 - ▶ 允許零長度字串索引



設計檢視

格式

- ➡ 格式是指資料顯示在螢幕上與列印出來的方式，也可直接在表單和報表物件上設定。
- ➡ 舉例來說，若設定格式為「百分比」，則欄位值「0.6」會顯示為「60%」。





設計檢視

輸入遮罩

- ➡ 遮罩是用來提醒或控制資料輸入的樣子，設定輸入遮罩時是以分號隔出以下三個區段：
 - ▶ 第一區段：遮罩本身。
 - ▶ 第二區段：「0」表示顯示在螢幕上(非使用者輸入)的符號也會被儲存在資料庫中；否則使用「1」。
 - ▶ 第三區段：提醒使用者輸入的定位字元。



設計檢視

- ▶ 譬如，若定義了「電話號碼」欄位，則可設定其輸入遮罩為「\ (999\) 0000\ -0000;0;#」。
- ▶ 這裡「9」代表數字或空格，「0」代表數字，而要顯示在螢幕上的符號如「(」，前面要加入反斜線「\」。





設計檢視

- ➡ 設定之後，使用者在輸入畫面上會看到「(###)####-####」。
- ➡ 若使用者輸入的資料為「(02)2462-2102」，則該資料可順利儲存在資料庫，但使用者無法輸入「(02)246-2102」，因為違反了規定的輸入數字個數。





設計檢視

預設值

- ➡ 使用者若未輸入任何值，則系統會為該欄位自動補上此值。譬如，我們可將「性別」欄位預設值設定為「F」。

驗證規則

- ➡ 如同之前「排名」欄位所示，我們可利用函數或算數運算子來寫出限制輸入值的式子。





設計檢視

- ▶ 譬如，我們可利用「like」函數，設定「電子郵件」欄位的驗證規則為「like `"*@"`」，這就表示任一個電子郵件都必須有「@」符號，但是前後可加入任意個數的字元。
- ▶ 在Access 2010裡也提供了一個【運算式建立器】的小工具，協助使用者定義複雜的運算式。





設計檢視

驗證文字

- ➡ 這裡是設定違反驗證規則時的提醒文字。
- ➡ 假設設定「電子郵件」欄位的驗證文字為「輸入資料不符規定」，當使用者輸入的「電子郵件」欄位值為「yahoo」時，則螢幕上會跳出一個小視窗，包含了文字「輸入資料不符規定」，以便讓使用者知道有錯誤的產生。





設計檢視

必須有資料

- ➡ 若填「是」的話，則使用者一定要在第一次輸入某筆資料列時，就提供該欄位的資料。為了增加建立資料時的彈性，通常是很重要的欄位，如主鍵，才會建議設定為「是」。

允許零長度字串

- ➡ 針對資料型態為「文字」或「備忘錄」的欄位，如果允許空字串，才可設定為「是」。



設計檢視

索引

- ➡ 索引是一種可加快搜尋資料的結構，指定「是」則Access會幫此欄位建立索引，不過根據欄位值在表格中是否重複出現的特性，還必須再指定為「可重複」或「不可重複」。





設定主鍵和外來鍵

- ➡ 首先必須進入【設計檢視】的畫面。以成績表格為例，由於該表格的主鍵包含「學號」和「課程」兩個屬性，所以必須按住CTRL鍵或Shift鍵，利用滑鼠左鍵同時選取這兩個屬性後，再按滑鼠右鍵。
- ➡ 接著在出現的清單中，選取【主索引鍵】，即完成設定。或者也可在選取完屬性後，點選功能表【資料表工具】中的【設計】頁面，然後在出現的功能表中點選【主索引鍵】選項。





設定主鍵和外來鍵

- ➡ 設定完主索引鍵之後，屬性的前面會出現如鑰匙般的小符號。注意Access使用「主索引鍵」這個詞來代表「主鍵」。
- ➡ 回到之前的Student表格，在【設計檢視】的視窗中，我們可將主索引鍵改設為「學號」欄位，此時Access預設的「識別碼」欄位不再是主鍵，我們即可以把它刪除。



設定主鍵和外來鍵

Microsoft Access 2010 介面顯示正在設計「Enroll」資料表。工具列中的「主索引鍵」按鈕（圖示為一把鎖）被選取。左側的「所有 Access 物件」清單中，「Enroll」資料表被選取。右側顯示「Enroll」資料表的設計視圖，包含欄位名稱和資料類型。

欄位名稱	資料類型
學號	簡短文字
課程	簡短文字
作業	數字
期中考	數字
期末考	數字
總成績	數字

設定主索引鍵





設定主鍵和外來鍵

- ▶ 若要設定成績表格的外來鍵，必須注意到相關表格都已經設定好主索引鍵，接著再開啟【資料庫關聯圖】。開啟的方式是，首先進入【資料庫視窗】，點選【資料庫工具】頁面中的【資料庫關聯圖】，如圖所示。



對應【資料庫關聯圖】的功能表





設定主鍵和外來鍵

- ▶ 點選之後會出現【顯示資料表】的視窗以便選取欲處理的表格，這時我們可選取學生表格「Student」和成績表格「Enroll」，執行之後的畫面如同下圖所示，注意到表格中的主索引鍵欄位前方會出現鑰匙符號。

資料庫關聯圖

Enroll	Student
學號	系別
課程	年級
作業	學號
期中考	姓名
期末考	地址
總成績	監護人
	排名

資料庫關聯圖





設定主鍵和外來鍵

➡ 以下列步驟設定表格欄位間的關聯：

1. 利用滑鼠指標，從學生表格的「學號」欄位拖曳到成績表格的「學號」欄位，此時出現【編輯關聯】的視窗。
2. 設定【強迫參考完整性】：設定之後就代表定義了外來鍵，在這裡我們將它選取。此項設定，會強制要求建立「成績」表格的學號時，該值必須已經存在於「學生」表格裡。





設定主鍵和外來鍵

3. 設定【串聯更新關聯欄位】：若是選取此項，則當我們更改學生表格的學號欄位時，會連帶更新成績表格的對應學號值，在這裡我們不選取。
4. 設定【串聯刪除關聯記錄】：若是選取此項，則當我們刪除學生表格的某筆紀錄時，會把成績表格中，所有學號相同的資料列連帶刪除，在這裡我們不選取。





設定主鍵和外來鍵

設定外來鍵(表格間的關聯)

編輯關聯

資料表/查詢(T): Student 關聯資料表/查詢(R): Enroll

學號	學號

☒ 強迫參考完整性(E)
☐ 串聯更新關聯欄位(U)
☐ 串聯刪除關聯記錄(D)

關聯類型: 一對多

確定
取消
連接類型(J)...
建立新的關聯(N)...





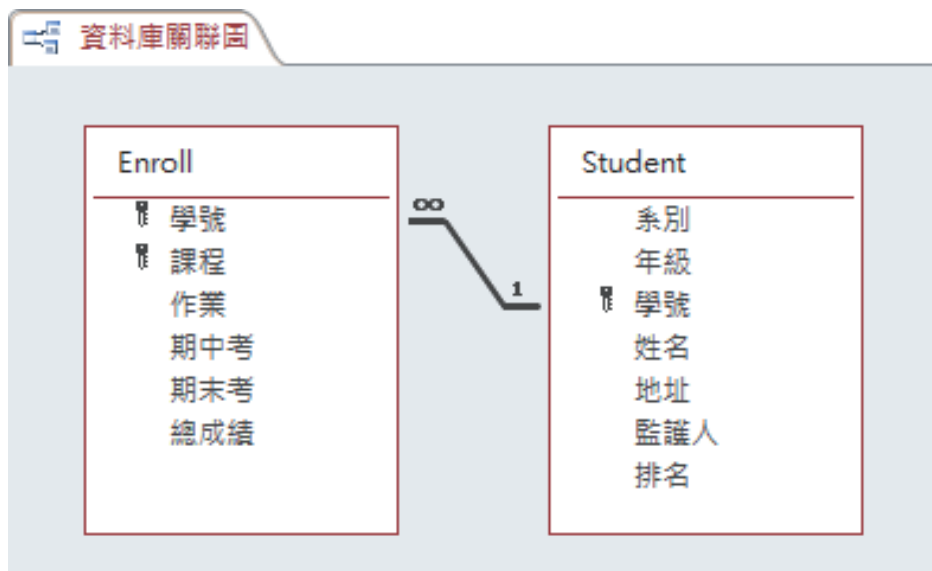
設定主鍵和外來鍵

- ➡ 透過之前已經設定好的主索引鍵定義，Access會自動偵測到這兩個表格「學號」屬性值的【關聯類型】是【1對多】，這表示學生資料表裡的一個學號，可以在成績表格裡出現多次。
- ➡ 按下【建立】，這時兩個表格間會出現一條線，把互相關聯的屬性連結起來，同時利用「1」和「∞」這兩個符號來表示「1對多」的關係。
- ➡ 最後記得關閉此工具時要將此圖儲存起來。





設定主鍵和外來鍵



完成關聯的設定





設定主鍵和外來鍵

- ➡ 針對資料表進行【設計檢視】時，點選功能表中【資料表工具】的【設計】頁面，也可看到【資料庫關聯圖】的選項。
- ➡ 點選之後功能表則會出現【關聯工具】的【設計】頁面，可供我們編輯既有的關聯或建立新的關聯。





建立SQL查詢

- ➡ Access以圖形化的介面為基礎，所以對於不熟悉SQL語法的使用者，提供了相當便利的工具。
- ➡ 如同之前提到的建立資料表方式，Access中也有數種不同建立查詢的方法，在【查詢】類型中，提供了【查詢精靈】和【查詢設計】兩種選項。





建立SQL查詢

- ➡ 首先，於【資料庫視窗】中的【建立】頁面中的【查詢】類型，選取【查詢設計】，則會出現【顯示資料表】視窗，其中顯示了資料庫中所有的資料表。
- ➡ 在該視窗中選取查詢句需要參考到的一個或數個資料表後，就會出現【查詢】視窗。





建立SQL查詢

- ➡ 該視窗的上半部會顯示已經選取好的資料表，而我們必須在視窗的下半部，或稱為【設計格線區】，指定輸出的欄位及提供限制資料選取的準則。
- ➡ 以第13-2節的SQL查詢為例，假設我們要建立查詢句1，如下所示：

```
SELECT 地址, 監護人  
FROM student  
WHERE 學號 = 'B9901'
```



建立SQL查詢

- ➡ 首先，必須在【顯示資料表】視窗中，選取要使用的資料表(對應到FROM子句)，這裡我們選取 **Student**。
- ➡ 進入【查詢】視窗後，先選取所有會使用到的欄位，在此查詢句中為「地址」、「監護人」和「學號」等三個欄位。
- ➡ 通常該欄位源自的資料表也會自動被選取，唯一要注意的情況是相同名稱的欄位出現在不同表格時，你必須明確指定要使用哪個表格的欄位。





建立SQL查詢

- ➡ 接著針對每個欄位，如果是要輸出的，也就是對應到SELECT子句，則必須要將【顯示】的方塊選取起來，通常該方塊會自動勾選，所以反過來必須把不需要輸出的欄位取消顯示。
- ➡ 如果是有限制條件的，也就是對應到WHERE子句，則必須在【準則】的空格裡指定該欄位的值所必須符合的限制式。
 - ▶ 以查詢句1而言，限制式是希望學號的欄位值等於某個常數，則直接輸入該常數即可。





建立SQL查詢

查詢句1的Access設計

Student

*
系別
年級
學號
姓名
地址
監護人
排名

欄位:	地址	監護人	學號
資料表:	Student	Student	Student
排序:			
顯示:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
準則:			"B9901"
或:			





建立SQL查詢

- ➡ 完成設計之後，按下右上方的，此時Access會詢問是否要儲存此查詢句，如果希望以後能繼續使用此設計，則回答【是】。
- ➡ 接著，在跳出的小視窗中輸入方便記憶的查詢句名稱(系統的預設名稱是【查詢】後面加上流水號)，此時我們不改變它，也就是維持名稱為【查詢1】，最後按下【確定】即可。





建立SQL查詢

- ▶ 若要查看此查詢句的執行結果，可以當還在設計查詢句時，點選功能表中【查詢工具】之【設計】頁面的【執行】選項。或是之後在【資料庫視窗】左側的窗格中，選取【查詢1】物件再按右鍵選取【開啟】，也同樣會執行該查詢句。



【查詢工具】功能表的【設計】頁面





建立SQL查詢

- ➡ 以查詢句3為範例，該查詢句重複列舉如下：

```
SELECT 學號, 姓名  
FROM student  
WHERE 排名<10 and 系別 = '資工系'
```

- ➡ 在這個查詢句中，由於「排名」欄位和「系別」欄位都有限制，所以必須將個別的限制式都在【準則】的空格裡輸入。





建立SQL查詢

- ➡ 由於排名的限制是要求欄位值小於某個常數，所以整個限制式必須連「<」符號一起表示。
- ➡ 值得注意的是，在Access裡，字串要以成對的雙引號括起來，雖然在上個範例中並沒有加入雙引號，但是Access會主動幫我們加入，所以不會有錯誤發生。





建立SQL查詢

查詢句3的Access設計

Student

+

系別
年級
學號
姓名
地址
監護人
排名

欄位:	學號	姓名	排名	系別
資料表:	Student	Student	Student	Student
排序:				
顯示:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
準則:			<10	"資工系"
或:				





建立SQL查詢

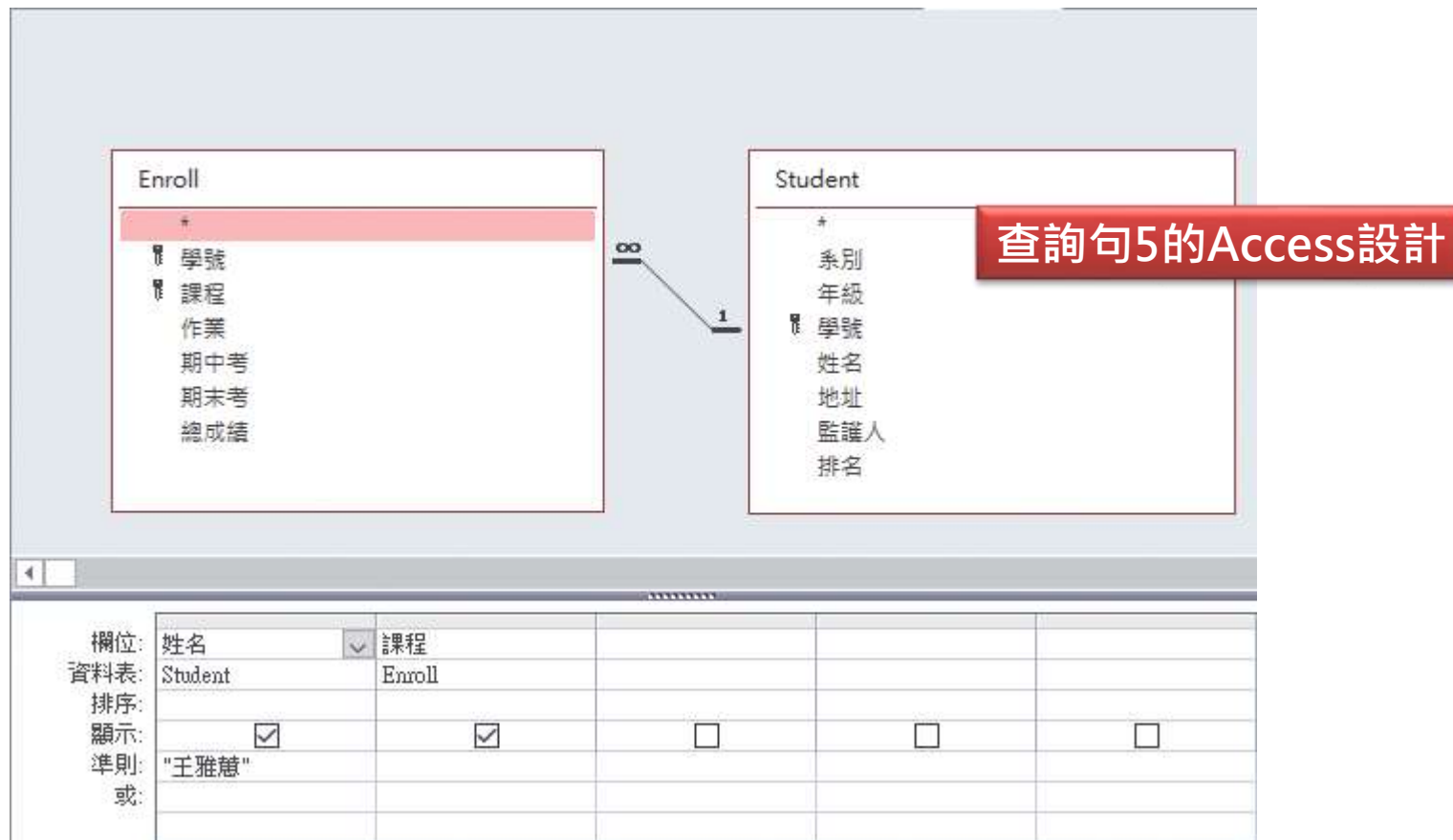
- 最後一個例子是對應到查詢句5，如下所列：

```
SELECT 姓名, 課程  
FROM student, enroll  
WHERE student.學號=enroll.學號 and  
       姓名 = '王雅蕙'
```

- 這裡必須將Student資料表和Enroll資料表一起選取進來。由於之前已經設定好兩個表格之間的關聯，所以對應的關聯圖會主動被包含進來，且連結限制「student.學號 = enroll.學號」就不用再次輸入。



建立SQL查詢





資訊科技專欄



Oracle簡介

Oracle資料庫軟體，在資料庫技術的研究上，一直具有領先的地位。其近年來最具代表性的應該首推1992年推出的Oracle 7。

該資料庫軟體不僅穩定、效率好，還提供相當多管理的工具，以及具主從架構(client-server architecture)的發展工具，所以一推出就很受好評。其自行研發的PL/SQL，也可供程式設計師直接利用SQL撰寫函數和程序。





13-4 資料探勘

- ➡ 資料探勘 (Data Mining ; DM) : 從大量的未處理資料 (raw data) 中 , 挖掘出有建設性的資訊 (information)
- ➡ 資料探勘的實例
 - ▶ 美國的沃爾瑪 (Wal-Mart) 公司 , 利用資料探勘的技術後 , 發現啤酒和尿片常常一起被購買
 - ▶ 公司把這兩項貨品擺在附近 , 由於顧客覺得購買方便 , 而造成這兩項產品的銷售率都成長3 成以上。





資料探勘的應用之一

- 針對客戶關係管理（Customer Relationship Management ; CRM）上，DM可能的應用與好處如下：
 - ▶ 了解客戶的滿意度，與客戶建立良好的互動；
 - ▶ 精確地掌握顧客消費習性，主動傳遞資訊給客戶；
 - ▶ 根據使用者的使用行為，事先偵察出不合理的使用狀況；
 - ▶ 分析潛藏的客戶群，或針對特定消費行為的顧客群做促銷。





資料探勘產生的常見資訊類別

➡ 關聯規則 (association rule)

- ▶ 規則範例：「若買啤酒則買尿片」，表示成：

啤酒 => 尿片

- ▶ 規則必須伴隨支持度 (support) 和信心度 (confidence)

➡ 分類 (classification)

- ▶ 根據既有的資料和一些特性，建立出一些分類法則，以推測一個新的狀況屬於哪一類：

申請者的年收入 > 100 萬 => 申請者的信用度 = 極佳；

申請者的年收入介於 100 萬和 60 萬間 => 申請者的信用度 = 好；

申請者的年收入 < 60 萬 => 申請者的信用度 = 普通；





資料探勘產生的常見資訊類別 (續)

分群 (clustering)

- ▶ 把具有相同或類似特性的物件分做同一群
- ▶ 在網路書店的範例中，我們可以根據使用者的購買行為將其分群，然後主動推薦新書
- ▶ 在空間資料上，我們可以把每一次犯罪的地點記錄下來，然後經由分群得知犯罪率特別高的區域





進行資料探勘的步驟

- ➡ 資料管理與準備
 - ▶ 資料可能必須轉化以符合資料探勘演算法的要求。
 - ▶ raw data 中不合理的資料必須移除
- ➡ 建立模型以分析資料
 - ▶ 選擇適當的演算法
 - ▶ 設定參數
- ➡ 輸出結果
 - ▶ 通常以圖形或報表呈現





13-5 XML簡介

- ➡ XML文件結構
- ➡ 文件物件模型
- ➡ 文件型態定義
- ➡ Xpath標準
- ➡ XQuery查詢語言





13-5 XML簡介

- ▶ **全球資訊網**(World-Wide-Web ; 簡稱WWW或Web)已經可以說是全世界資訊分享的主要方式，因為它在無遠弗屆的廣域網路環境中，提供一種便利且簡單的方式去存取資料，所以相當多的企業已經將其產品廣告或可分享的資料放在網際網路上。





13-5 XML簡介

- ▶ WWW上所接受的資料型態是符合HTML格式的文件，但是原本HTML是設計為顯示資料之用，以便將文件內容呈現在使用者面前，所以內含許多控制輸出的標籤 (tag)，如 `<table>`、``、``、``...等，而不是表示資訊的內容及它的結構，其缺乏對資訊意涵的描述，所以不利於自動化的資訊傳遞與交流。





13-5 XML簡介

- ▶ **可延伸式標記語言** (Extensible Markup Language ; XML) , 已成為最近Web上相當受到重視的格式。
- ▶ XML是由W3C制定的一個有關於描述資訊的上層語言(meta language) , 其1.0版於1998年2月正式推出。
- ▶ XML的目的為定義一個描述資料之標準 , 允許使用者可以自由地定義標籤 , 以適當的結構來描述所要傳輸的資料。





13-5 XML簡介

- ➡ XML規格將資料與使用者介面分離，所以易於達到自動化處理的目的。
- ➡ XML對於各類型資料(如物件、文章、圖形、文字檔、二元檔等)都能標註，且以文字為基礎來表示資料，不僅容易在異質系統之間傳遞交流，且能穿過防火牆，便於在不同企業間進行資料交換。
- ➡ 所以，已經有相當多組織，將其資料表示標準化，以XML格式表示，作為在Web上資料共享的主要依據。





13-5 XML簡介

- ➡ 從XML提出至今，幾家重要的軟體廠商，如Oracle、HP、IBM、Microsoft等，皆全力的投入研發，並配合W3C的規格，不斷地推出新的產品與應用，所以XML的重要性絕對是不可忽視。
- ➡ 由於XML只提供表示資料的方式，必須配合W3C另外定義的輔助技術來處理XML資料。





XML文件結構

- ➡ 一個XML文件的範例如圖13-18所示，它表示了三本書籍的資料，每一本書則分別描述了書名、作者、出版廠商、出版日期等訊息。
- ➡ 由範例中，可看出XML文件利用適當的標註，來提供資料的結構及與語意有關的資訊。
- ➡ 特別注意的是：該文件只表示「資料」，並無指定「顯示介面」。





XML文件結構

XML文件範例

```
L1 <Books amount="3">
L2 <Book>
L3   <Title>Essential XML</Title>
L4   <Authors>
L5     <Author>Box</Author>
L6     <Author>Skonnard</Author>
L7     <Author>Lam</Author>
L8   </Authors>
L9   <Publisher>AW</Publisher>
L10  <Date year="2000" month="7"/>
L11 </Book>
L12 <Book>
L13   <Title>計算機概論 </Title>
L14   <Authors>
L15     <Author>趙坤茂</Author >
L16     <Author >張雅惠</Author>
L17     <Author >黃寶瑩</Author >
L18   </Authors>
L19   <Publisher>全華</Publisher>
L20   <Date year="2004" month="7"/>
L21 </Book>
L22 <Book>
L23   <Title>Spanning Trees and Optimization Problems</Title>
L24   <Authors>
L25     <Author>吳邦一</Author>
L26     <Author>趙坤茂</Author>
L27   </Authors>
L28   <Publisher>Chapman & Hall/CRC Press, USA. </Publisher>
L29   <Date year=" 2004" month="1"/>
L30 </Book>
L31 </Books>
```





XML文件結構

- ➡ 以下利用此範例進一步說明XML文件的架構。XML可說是由一個個元素(element)所組成的。
- ➡ 所謂的元素，就是由一個開始標籤(start-tag)到對應的結束標籤(end-tag)為止，包含其中的所有內容。
- ➡ 譬如，從L2行的開始標籤 `<Book>` 到L11行的 `</Book>` 為止，表示了一個Book元素。





XML文件結構

- ➡ 元素中可以包含其他元素，稱作子元素，譬如，該Book元素有四個子元素：Title、Authors、Publisher、Date。
- ➡ XML要求文件必須格式正確(well-formed)，也就是每個XML文件中只能有一個在最外層的根元素(root element)，如L1-L31行的Books元素；同時，每個元素的開始標籤與結束標籤須成對，標籤之間不可交錯，即所有元素的排列必須為嚴謹的巢狀結構。





XML文件結構

- ➡ 元素可包含屬性(attribute)，所有的屬性之值必須加上單引號或雙引號，如L10行的Date元素包含一個屬性year，其值為2000。





文件物件模型

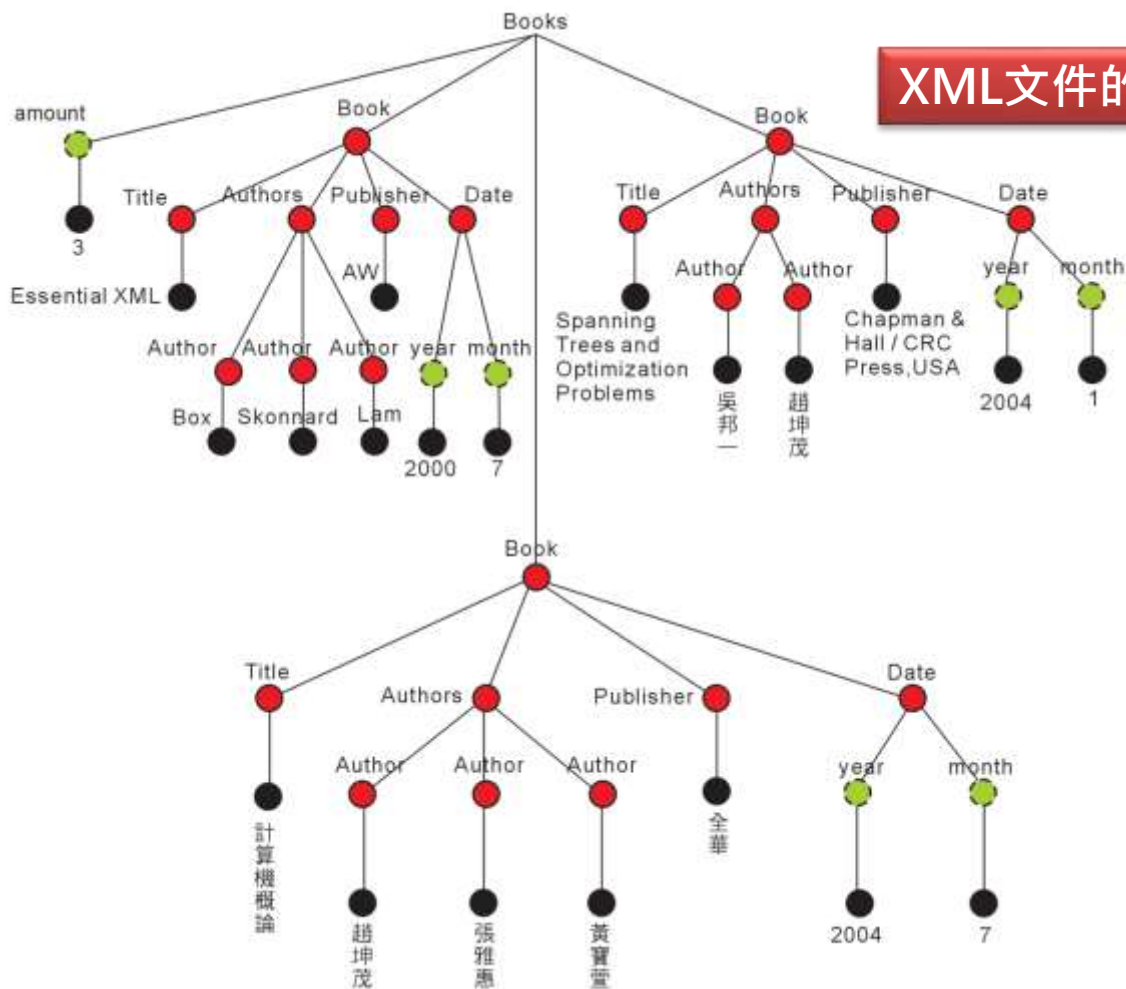
- ➡ 文件物件模型 (Document Object Model ; DOM)，是W3C定義來描述XML文件的架構，同時規範存取XML資料的介面，然後各家廠商可以根據該標準介面，自行提供實作的細節。
- ➡ DOM的基本觀念，就是將XML檔案分解成個別的元素、屬性等，然後以它們為節點，表示成一個有順序的標籤樹(ordered label tree)。





文件物件模型

XML文件的DOM樹狀表示





文件型態定義

- ➡ 相較於結構化的關聯式資料庫，XML提供了一種半結構化的 (semi-structured) 表示資料的方式。
 - ▶ 舉例來說，關聯式資料庫的每筆資料都必須具有固定個數的欄位，但是XML資料內某一個元素可以出現一次以上。
- ➡ 這些資料定義規格，可以利用W3C所頒訂的**文件型態定義** (Document Type Definition ; DTD) 來描述。





文件型態定義

- ▶ 該定義可規範特定XML文件的格式，也就是將每一個元素可以包含哪些子元素或屬性、各元素出現的順序等，清楚地加以定義和規範。

L1	<?xml version="1.0" encoding="Big5"?>
L2	<!ELEMENT Books (Book*)>
L3	<!ATTLIST Books amount NMTOKEN "1">
L4	<!ELEMENT Book (Title, Authors, Publisher, Date)>
L5	<!ELEMENT Title (#PCDATA)>
L6	<!ELEMENT Authors (Author*)>
L7	<!ELEMENT Publisher (#PCDATA)>
L8	<!ELEMENT Date EMPTY>
L9	<!ATTLIST Date year #CDATA #REQUIRED month #CDATA #REQUIRED>
L10	<!ELEMENT Author (#PCDATA)>

DTD範例





文件型態定義

- ➡ 在DTD文件中，ELEMENT標籤之後定義的是元素的名稱，接著用小括號括起來的，是該元素的「內容模型」，也就是在對應的XML文件中可以出現的內容。
- ➡ 在L4行的定義，規定了Book這個元素可包含Title、Authors等四個子元素。
- ➡ 至於L3行的「NMTOKEN」則是規範屬性值內不能包含空白。





文件型態定義

- ➡ 在L5行中進一步註明Title元素存放的資料為「#PCDATA」(Parsable Character Data)，該字串為預先定義的標記，代表可解析的文字資料。
- ➡ ATTLIST標籤則是用以宣告元素的屬性，包含了屬性名稱、屬性類別及預設行為的描述，若屬性不只一個時，可以用這三個部分為一個單位一直重複下去。





文件型態定義

- ➡ L9行定義了Date這個元素具有year和month這二個屬性。其中，year屬性的類別為 #CDATA，表示該屬性值為一般文字；預設行為的描述為 #REQUIRED，表示該屬性值一定要存在。
- ➡ 特別注意的是：DTD 允許類似 regular expression 的符號，如 L2 行的星號 (*) 代表 Books 元素裡可包含多個 Book 子元素。





XPath標準

- ➡ XPath是節點位置語言，用來取出位於特定位置的XML元素。
- ➡ 我們直接參照圖13-19的樹狀結構，可以更清楚地瞭解XPath的寫法。
- ➡ 在XPath中，我們必須指定如何從一個節點走到下一個節點，最常見的是把從根元素到該節點的完整路徑寫出來，如「/Books/Book/Title」。





XPath標準

- ➡ 該XPath敘述式，會回傳下列三個元素：

```
<Title>Essential XML</Title>  
<Title>計算機概論</Title>  
<Title>Spanning Trees and Optimization Problems</Title>
```

- ➡ 有時我們不是很確定元素在樹裡的確切位置，則我們可以在節點間使用「//」符號。
 - ▶ 在XPath的標準中，「/」代表元素間具有父子關係，而「//」則代表元素間具有祖孫關係。





XPath標準

- ➡ 假設我們不確定Title元素的完整路徑，可以下「`//Title`」或「`/Books//Title`」，其回傳的元素會跟上式「`/Books/Book/Title`」相同。
- ➡ 可以利用萬用字元，取出不限定名稱的所有元素，如「`/Books/Book/*`」。
- ➡ 也可針對某個節點的內容加以限制。





XPath標準

- ➡ 「/Books/Book[//Author= “趙坤茂”]/Title」，會回傳所有「趙坤茂」寫的書本的標題。
- ➡ XPath標準裡也提供許多函數以便使用者下達複雜的限制式，譬如表示式「//*[count(Book)=3]」，則會指出擁有三個「Book」子元素的所有元素。





XQuery查詢語言

- 由於預期XML資料在未來將大量的出現，XML資料查詢的議題會相當重要，所以，W3C在2001年的2月首先提出了XML Query Requirements，討論對XML資料做查詢時的需求，隨後也在2001年的6月提出了XQuery 1.0 Working Draft，以作為XML查詢語言的標準。經過多年的修改與討論，XQuery 1.0的正式版本終於在2007年1月頒佈。





XQuery查詢語言

- ➡ XQuery基本上利用XPath的語法，在XML資料中找尋出符合特定路徑的資料。
- ➡ 為了可以更詳盡的描述不同路徑資料間的關係，XQuery也設計成由很多不同功能的子句所構成。
- ➡ 基本上來說，一個XQuery的敘述式為一個FLWOR表示式，包含了for、let、where、order by、return五個部分。





XQuery查詢語言

- ➡ **for**子句類似SQL的FROM子句，其主要功能是宣告變數，並指定對應的元素集合，以便遞迴取得(iterate over)一個特定的元素加以處理。
- ➡ **let**子句則是將變數直接與某一個表示式結合(binding)，可以簡化之後查詢句的敘述。
- ➡ **where**和**order by**則分別類似SQL的WHERE子句和ORDER BY子句，前者允許對變數做條件的限制，後者則可以對輸出的元素值指定排序的方式。





XQuery查詢語言

- **return**則可以建構新的XML元素為查詢的輸出，類似SQL的SELECT子句。

L1	for \$b in document("http://dblab.cs.ntou.edu.tw/B0001.xml")
L2	/Books/Book
L3	where contains(\$b/Title/text(), "XML")
L4	return
L5	<Results>
L6	{ \$b }
	</Results>

XQuery範例 —— 找出Title元素值包含XML的書籍資料





XQuery查詢語言

- ▶ 以圖13-22的範例來說明如何撰寫一個XQuery的查詢句，該查詢句是要找出書的標題內包含「XML」字串的書籍資料，假設XML資料是存放在網址「<http://dblab.cs.ntou.edu.tw>」下檔名為「B0001.xml」的文件內。





XQuery查詢語言

- ➡ L1 行指定「B0001.xml」檔案內每一個位於「/Books/Book」的元素給\$b這個變數，我們可以同時指出文件的所在位置(URL)為「<http://dblab.cs.ntou.edu.tw/>」。
- ➡ L2行的目的在限定 \$b元素下的Title子元素必須包含「XML」這個字串，所使用到的函數 **contains**，是用來檢查元素的內容值是否包含某個特定字串。





XQuery查詢語言

- ➡ L3行之後的表示式是將符合條件的查詢結果傳回來，也就是印出「/Books/Book」的內容。
- ➡ L4行及L6行指定輸出之XML資料的根節點，其開始標籤為 <Results>，而其結尾標籤為 </Results>。





XQuery查詢語言

```
L1    <Results>
L2      <Book>
L3        <Title>Essential XML</Title>
L4        <Authors>
L5          <Author>Box</Author>
L6          <Author>Skonnard</Author>
L7          <Author>Lam</Author>
L8        </Authors>
L9        <Publisher>AW</Publisher>
L10       <Date year="2000" month="7"/>
L11     </Book>
L12   </Results>
```

對應於圖13-21之XQuery的查詢結果





XQuery查詢語言

- ➡ 圖13-24是另一個XQuery的範例，這裡我們希望以作者「趙坤茂」為主，把他寫的書的所有資料都列出來，不過要以書名排序。這裡的寫法和之前的查詢句有幾個差異。
 - ▶ 首先，可以把輸出文件的根元素，寫在整個查詢句的最外頭。
 - ▶ 另外，內容值的限制可以直接用等式。
 - ▶ 最後，**order by**子句是在**return**子句之前。輸出的結果如圖13-25所示。





XQuery查詢語言

```
L1 <Books>
L2 {
L3   for $a in /Books/Book, $c in $a //Author
L4   where $c = "趙坤茂"
L5   order by $a/Title
L6   return $a
L7 }
L8 </Books>
```

XQuery範例 —— 找出作者趙坤茂所寫的所有書籍





XQuery查詢語言

```
L1  <Books>
L2    <Book>
L3      <Title>Spanning Trees and Optimization Problems</Title>
L4      <Authors>
L5        <Author>吳邦一</Author>
L6        <Author>趙坤茂</Author>
L7      </Authors>
L8      <Publisher>Chapman & Hall/CRC Press, USA. </Publisher>
L9      <Date year=" 2004" month="1"/>
L10   </Book>
L11   <Book>
L12     <Title>計算機概論 </Title>
L13     <Authors>
L14       <author>趙坤茂</author>
L15       <author>張雅惠</author>
L16       <author>黃寶萱</author>
L17     </Authors>
L18     <Publisher>全華</Publisher>
L19     <Date year="2004" month="7"/>
L20   </Book>
L21 </Books>
```

