

第二章 資料模式





本章內容

- 2.1 資料模式簡介
- 2.2 階層式資料模式
- 2.3 網路式資料模式
- 2.4 關聯式資料模式
- *2.5 物件導向式資料模式
- *2.6 XML 資料模式



資料模式簡介

- 資料模式：描述資料庫或資訊系統的資料表示方式、表示規範與運用方式的抽象概念
- 完整的資料模式由以下三者所定義：
 - 資料結構 (Data Structures) — 資料在資料庫中的表示方式。
 - 整合限制條件 (Integrity Constraints) — 資料在結構與表示方式中的合法條件。
 - 資料的運算 (Data Manipulation 或 Operations) — 如何在資料結構上對資料做運算。



資料模式的演進

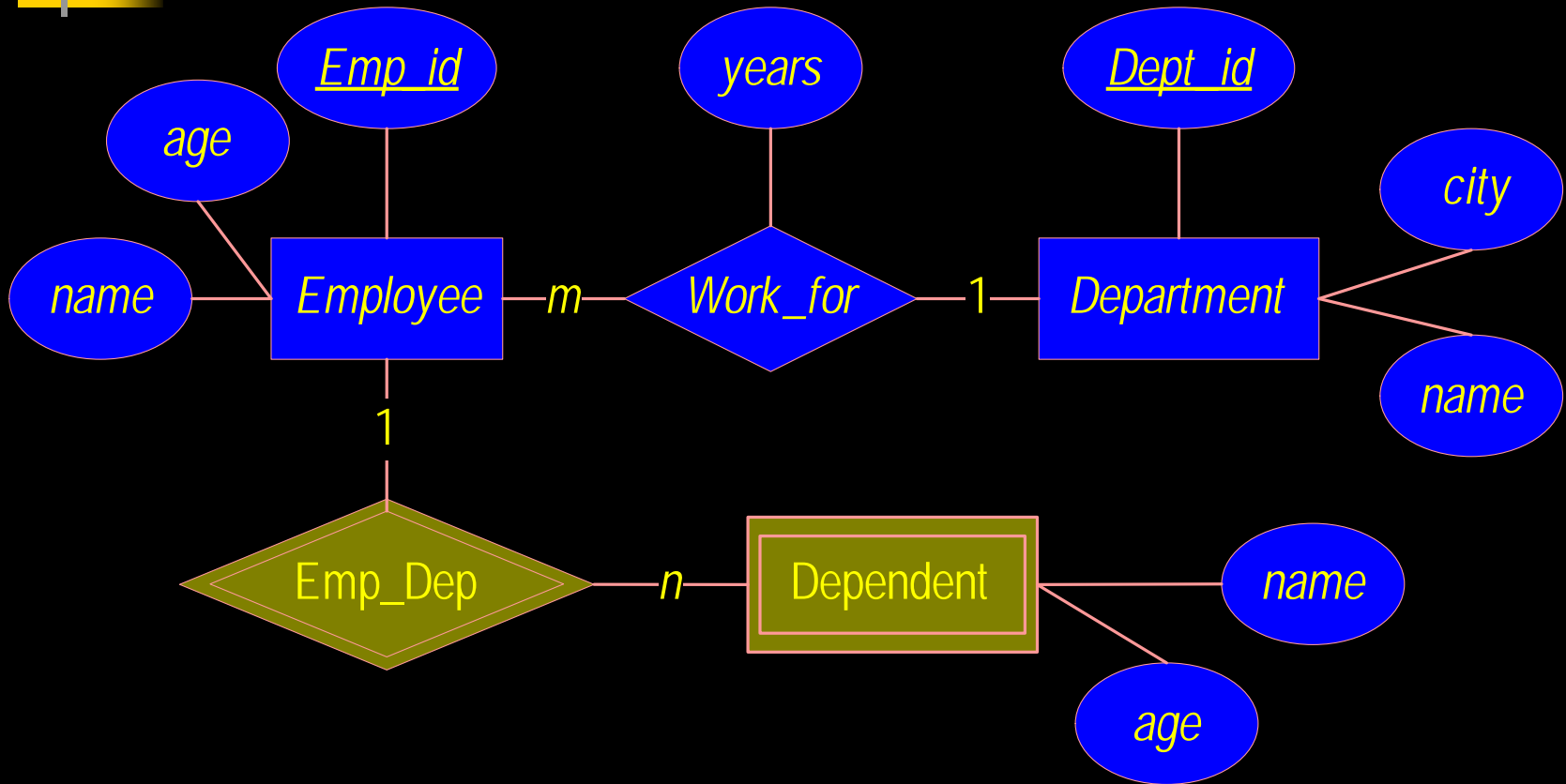
- 階層式資料模式 (Hierarchical Data Model)
- 網路式資料模式 (Network Data Model)
- 關聯式資料模式 (Relational Data Model)
- (中間經過許多的演變與發展：E-R Model, Functional Model, Semantic Data Model, E²-R...)
- 物件導向式資料模式 (Object-Oriented/Object-Relational Data Model)



個體-關係模式 (E-R Model)

- Prof. Peter P.S. Chen 於 1976 年提出
- 將資料組成分成兩大類：
 - 「個體」 (Entities)
 - 「關係」 (Relationships)。
- 並沒有定義三大要素中的資料運算部份。
- 常常拿來應用在邏輯資料庫的綱要設計 (Logical Database Design) 上 (詳見本書 5.8 節及第九章)

個體-關係圖 (E-R Diagram)



Weak Relationship

Weak Entity (詳見本書 5.8 節及第九章)



語意資料模式 SDM

- M.M. Hammer & D.J. McLeod 1978 年提出
- 也沒有定義資料的運算部份
- 應用於邏輯資料庫的綱要設計
- 或是人工智慧處理的內部資料表示法上



函數資料模式 (Functional Model)

- D. Shipman 於 1981 年所提出
- 將資料組成分成兩大類：
 - 「個體」 (Entities)
 - 「函數」 (Functions)
- 函數的定義域與值域都是由個體所組成
- 函數代表了個體與個體之間的關係
- 採用函數為導向的語言 DAPLEX



擴充後的個體-關係模式 (EER)

- T.J. Teorey, D. Yang, and J.P. Fry (1986) 提出 Extended E-R Model (或 Enhanced E-R, E²-R)
- 主要還是應用於邏輯資料庫的綱要設計上
- 這些觀念後來還應用在物件導向式資料模式上
- 個體-關係模式的影響相當深遠
- 高普考曾經考過此一問題



階層式資料模式

- 先有 (階層式) 資料庫系統，如：IBM 的 IMS/VS，爾後才有 (關聯式) 資料模式
- 階層式資料模式是後來才訂定下列各項
 - 階層式資料模式的資料結構
 - 階層式資料模式的整合限制條件
 - 階層式資料模式上的資料運算
- 適合用來描述一對多的關係，但不適合多對多



階層式資料模式的資料結構

- 一個有序集合 (Ordered Set)
- 由可以重複同一種組織的樹狀結構所組成
- 由「樹根」(Root) 資料記錄以及一個有序的「子樹」(Subtree) 集合所構成
- 由各種資料記錄以階層方式排列而成
- 完整順序是依內部節點由上至下、由左至右來定義



階層式出版商資料庫綱要

PUBLISHER

Name	Address
------	---------

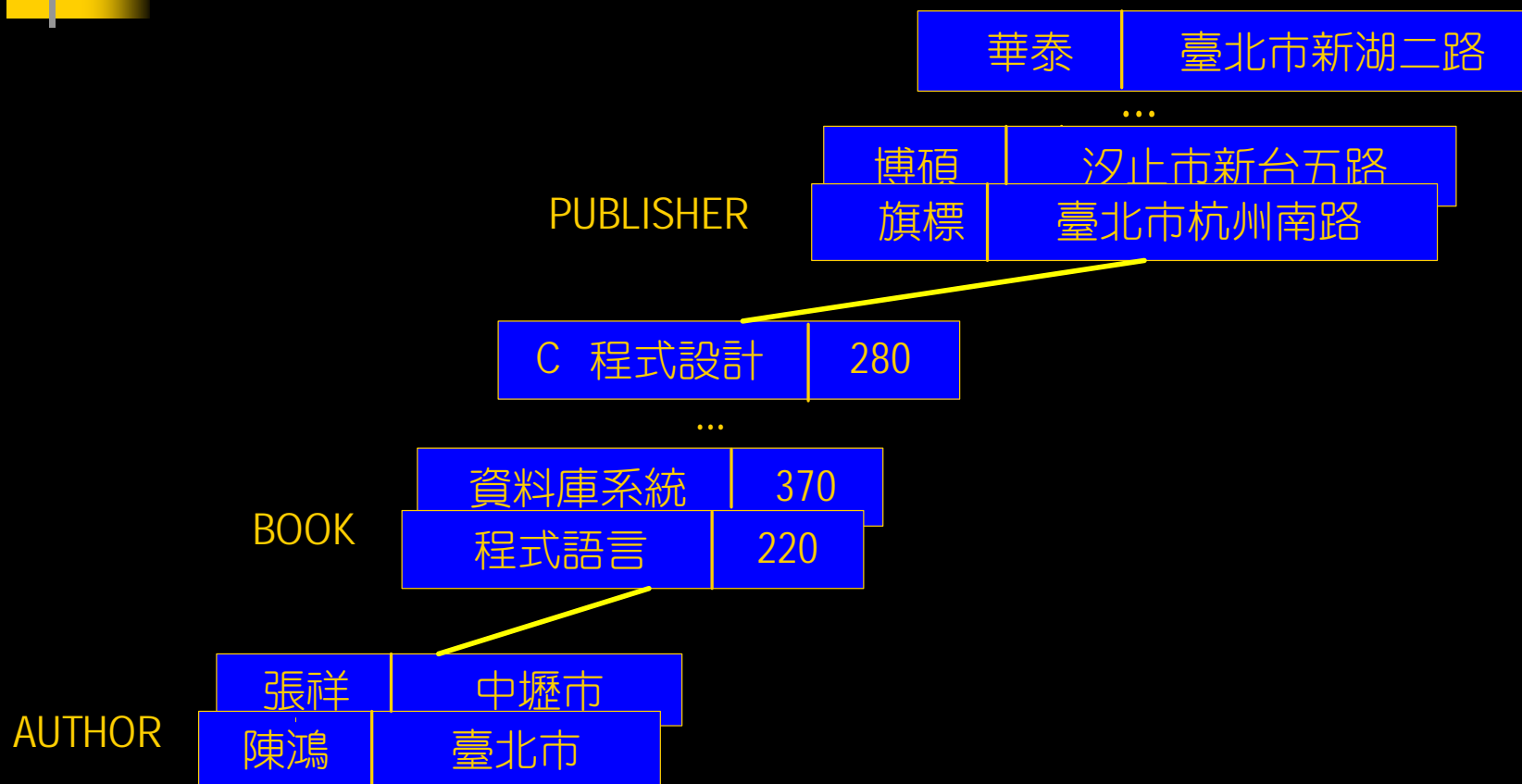
BOOK

Bookname	Price
----------	-------

AUTHOR

Name	City
------	------

出版商資料庫的樹狀結構資料





階層式模式中的整合限制條件

- 階層式模式的整合限制條件 (僅有一條) :
任何的「孩子節點」 (Child Node) 都必須要有
「父親節點」 (Parent Node) 的存在。
 - 只是定義資料結構上的最基本部份
 - 可以依照目標資料庫的意義予以擴充，漸次增加該
資料庫的整合限制條件



階層式資料模式上的運算

- 就是樹狀結構 (Tree Structure) 上的運算：
 - 在資料庫中找到某棵特定的樹狀結構。
 - 在資料庫中從特定的樹往下一棵樹移動。
 - 在特定樹中的資料記錄中移動或搜尋。
 - 在不違反整合限制條件的情況下，加入/刪除/更新一筆資料記錄。
 - 在不違反整合限制條件的情況下，刪除一整棵樹狀結構，包含其下的所有節點。

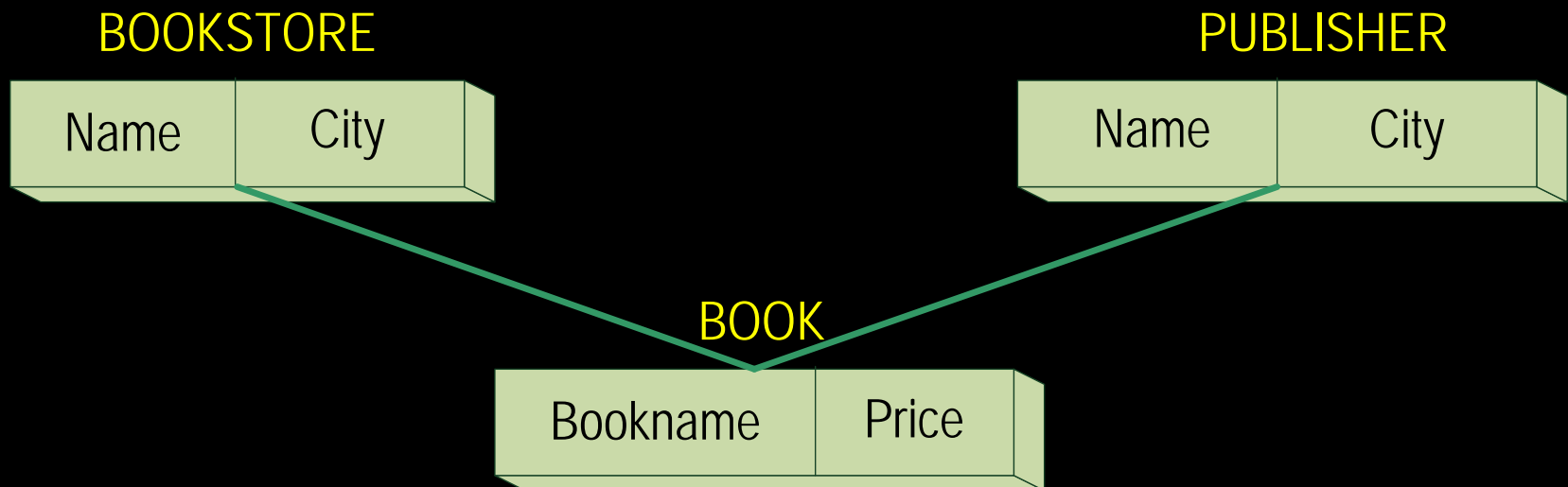


網路式資料模式

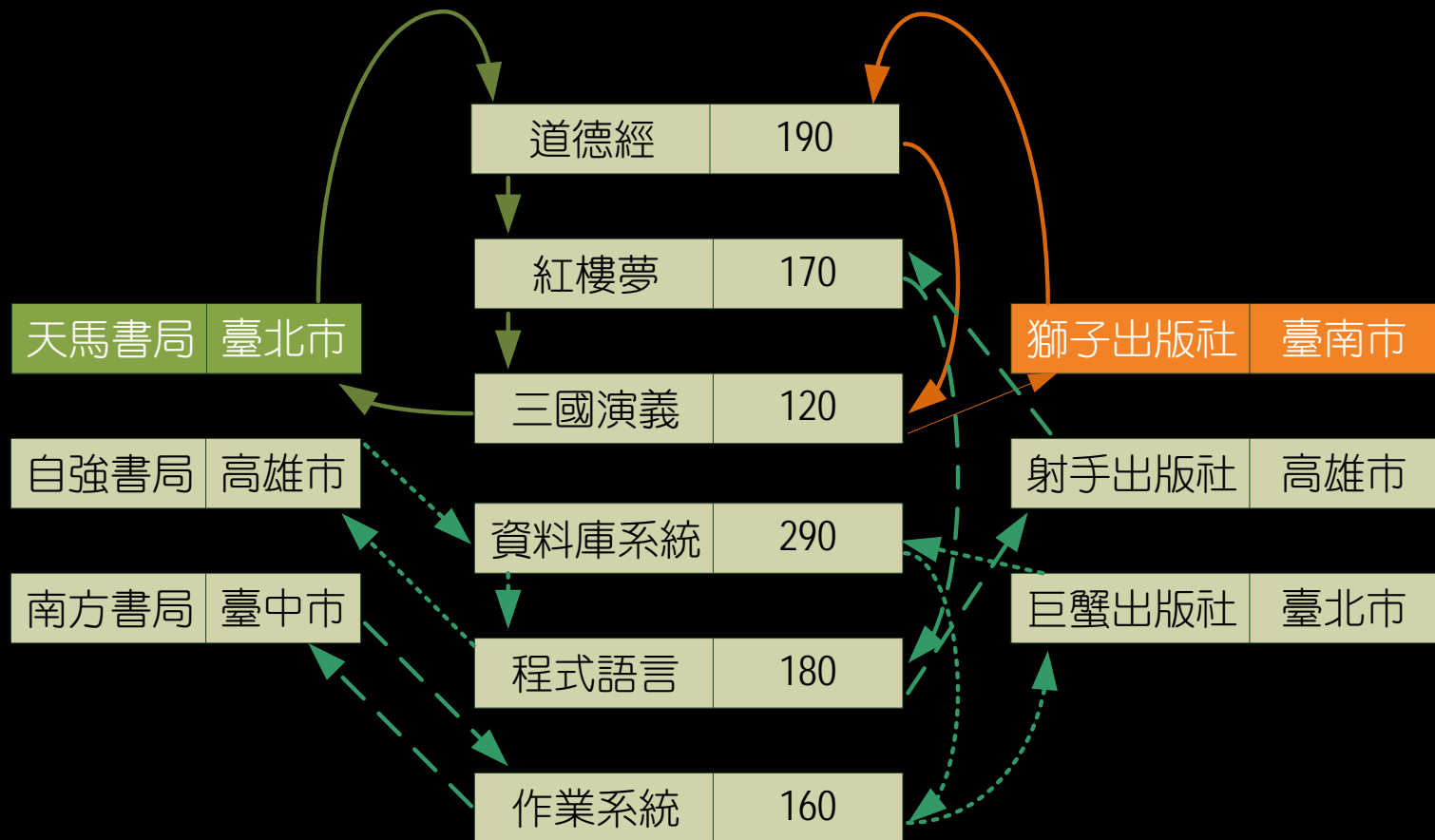
- 適合描述多對多的關係
- CODASYL (Conference on Data Systems Language) 的 Data Base Task Group (DBTG) 在 1971 年提出
- 先有網路式資料庫系統，爾後才訂出網路式資料模式

網路資料模式的資料結構

- 一個「孩子節點」可以有數個「父親結點」
- 由資料記錄的集合 (A Set of Records) 與鏈結集合 (A Set of Links) 所構成
- 綱要範例



網路式資料範例





網路式資料模式上的 整合限制條件

- 限制條件只有一條：除非其父親節點已經存在了，否則任何的子節點都不能加入資料結構中。
 - 只定義資料結構上的最基本部份
 - 可以依照目標資料庫的意義予以擴充，漸次增加該資料庫的整合限制條件

網路式資料模式上 的資料運算

- 尋找某一筆特定的資料記錄。
- 從父節點根據某鏈結，移往上一孩子節點。
- 根據某個鏈結，從某個孩子節點移往下一個孩子節點動。
- 從某個孩子節點根據某個鏈結，往它的父親節點移動。
- 在不違反整合限制條件的情況下，新增/刪除/修改一筆記錄。



關聯式資料模式

- E. F. Codd 在 1970 年時所提出
- 以數學上的「集合論」作為理論基礎
- 將資料以「關聯式」的表格方式來表示
- 「關聯表」(Relations) 為處理的單位 (集合)
- 不是以單一筆的記錄做為處理單位

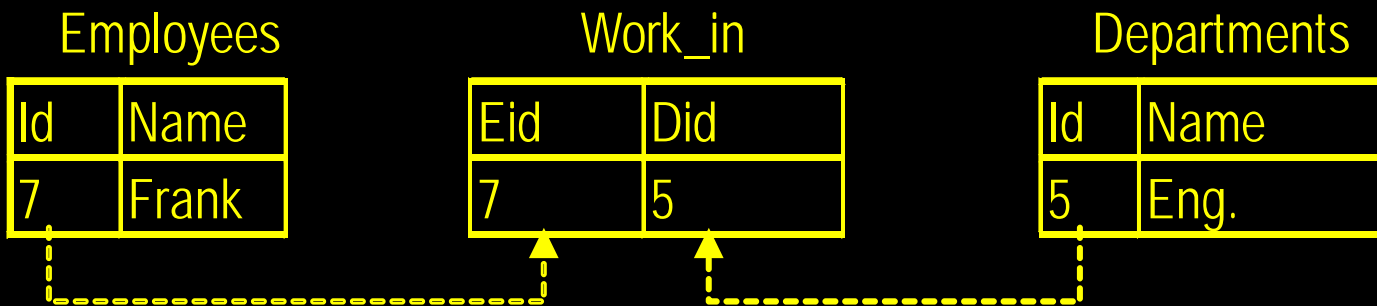


題外話：集合論中的詭論 (Paradox)

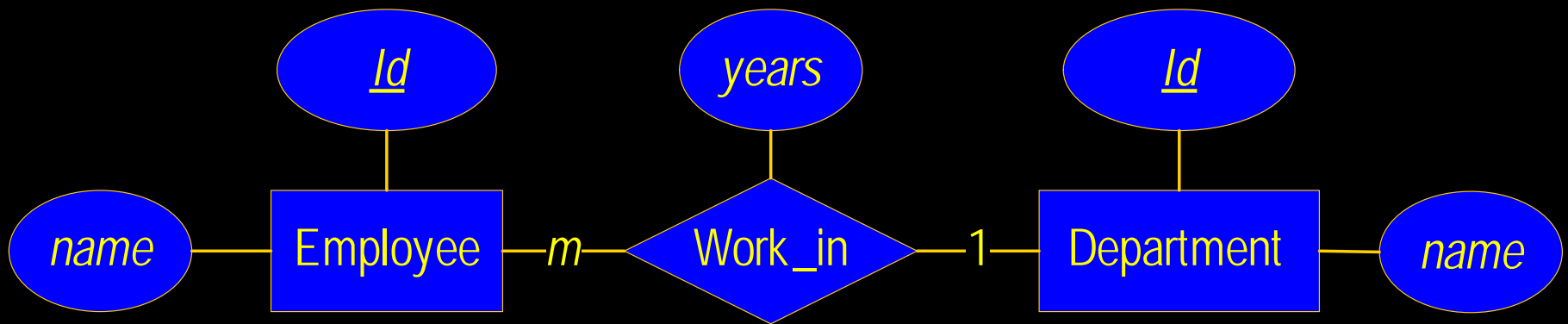
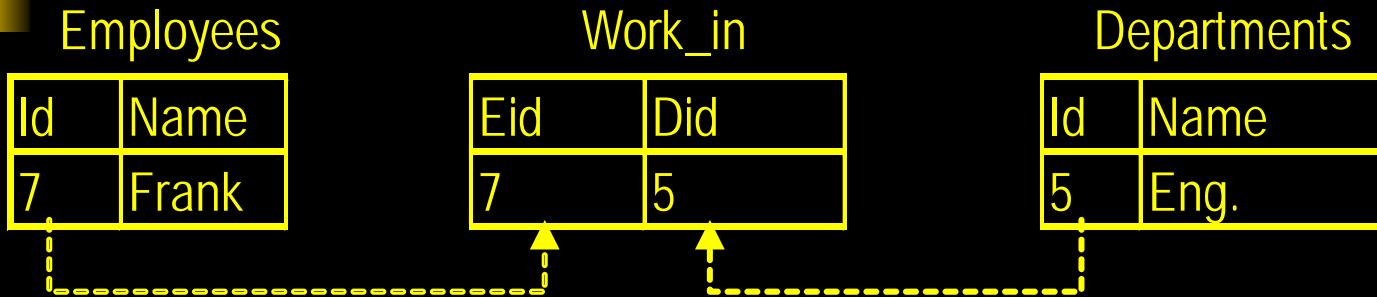
- 所有集合所成的集合是否仍是一個集合？錯！
為什麼？無窮集合也是集合！
對！但 A 會變成屬於 A 中的元素？
- 理髮師只替那些不替自己理髮的人理髮，那他自己替不替自己理髮？**很矛盾!**
- 說實話者槍斃，說謊者吊死，則回答：「我是來給你吊死的！」該槍斃或吊死？**怎麼做都不對!**

關聯式資料關係連結法

- 資料之間的關係以資料值 (Data Value) 連結
- 不像「階層式資料模式」與「網路式資料模式」以指標 (Pointer) 作為連結資料之間的關係
- 資料間的關係以合併運算 (Join) 完成



個體-關係模式的資料庫設計





關聯式資料模式

- (1) 資料的表示法：表格式 (關聯表) 資料結構
- 將在第四章中詳細探討

Books

<i>id</i>	<i>bookname</i>	<i>author</i>	<i>price</i>	<i>publisher</i>
1	三國演義	羅貫中	120	古文出版社
2	水滸傳	施耐庵	170	中庸出版社
3	紅樓夢	曹雪芹	170	春秋出版社
4	西遊記	吳承恩	140	聊齋出版社
5	水經注	酈道元	120	易經出版社
6	道德經	老子	190	大唐出版社

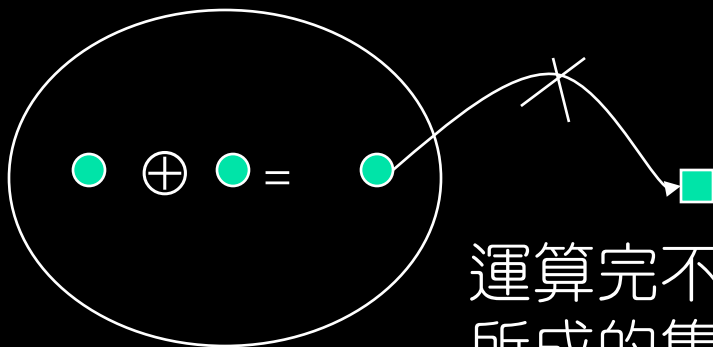


關聯式資料模式 (續)

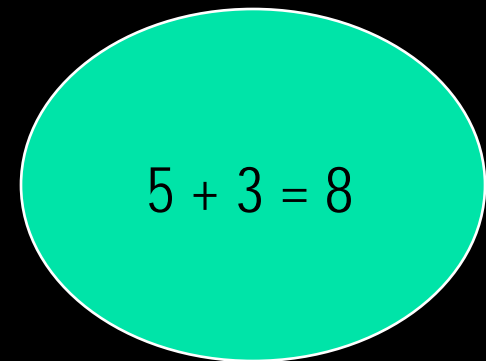
- (2) 資料表示法上的整合限制條件：
 - 「個體整合限制」 (Entity Integrity) 一定義了關聯表內部的限制條件
 - 「參考整合限制」 (Referential Integrity) 一定義了關聯表彼此之間的限制條件
- 將在第五章中詳細探討

關聯式資料模式 (續)

- 「封閉性」 (Closure)：一個集合中的元素運算完後所得到的結果仍在該集合中，稱為封閉性，例如：整數 + 整數 = 整數，不會產生其他不屬於整數集合的值，那麼我們說 '+' 對於整數集合有封閉性
- 對關聯表做運算後，所得到的結果或中間結果也都是關聯表



運算完不會跑出關聯表所成的集合之外





關聯式資料模式 (續)

- (3) 作用於資料表示法上的運算—兩套完整的運算模式：
 - 關聯式代數 (Relational Algebra) —第六章說明
 - 關聯式計算 (Relational Calculus) —第六章說明
 - 兩者具有同等的運算能力。
 - SQL 完美地融合了兩套運算模式—第七章說明



關聯式模式的不足

- 關聯式模式無法滿足複雜的應用：
 - 多媒體的辦公室自動化環境 (Multimedia Office Information Systems)
 - 電腦輔助設計上的應用 (Computer Aided Design) — 如：建築、VLSI、機械製造上的電腦輔助設計 (Computer Aided Manufacturing)
 - 地理資訊系統 (Geographic Information Systems)
 - 電腦輔助軟體工程上的 CASE tools (Computer Aided Software Engineering Tools)
- 大多數關聯式資料庫管理系統在大型物件 (如：影像資料或是大量文件資料) 的支援上，大多採用 **Binary Large OBjects (BLOBs)** 或是 **Character Large OBjects (CLOBs)** 來儲存，無法直接用查詢語言分析內容，在查詢效能上很不理想



複雜應用共同的需求

- 複雜應用都有底下幾個共同的需求與特性：
 - 層層包覆 (Objects Composed of Other Objects) 的複雜結構關係
 - 需要有多重值的屬性 (Multivalued Attributes)，或是可以儲存集合的屬性 (Set-Valued Attributes)
 - 資料有縱向分類 (Generalization and Specialization)、橫向組合 (Aggregation) 與繼承關係 (Inheritance)
 - 具有時間 (Temporal) 與空間 (Spatial) 上的屬性與對應關係
 - 不同版本的需求 (Multi-Version Control)

過渡時期所提出的 Nested Relation 結構表示法

- Create type **address**(street char(20), city char(20), state char(20), postcode int)
- Create type **customer**(name char(20), address **Address**)
- Create type Person(order_id int, date datetime, customer **customer**, phone char(10))

一筆
記錄

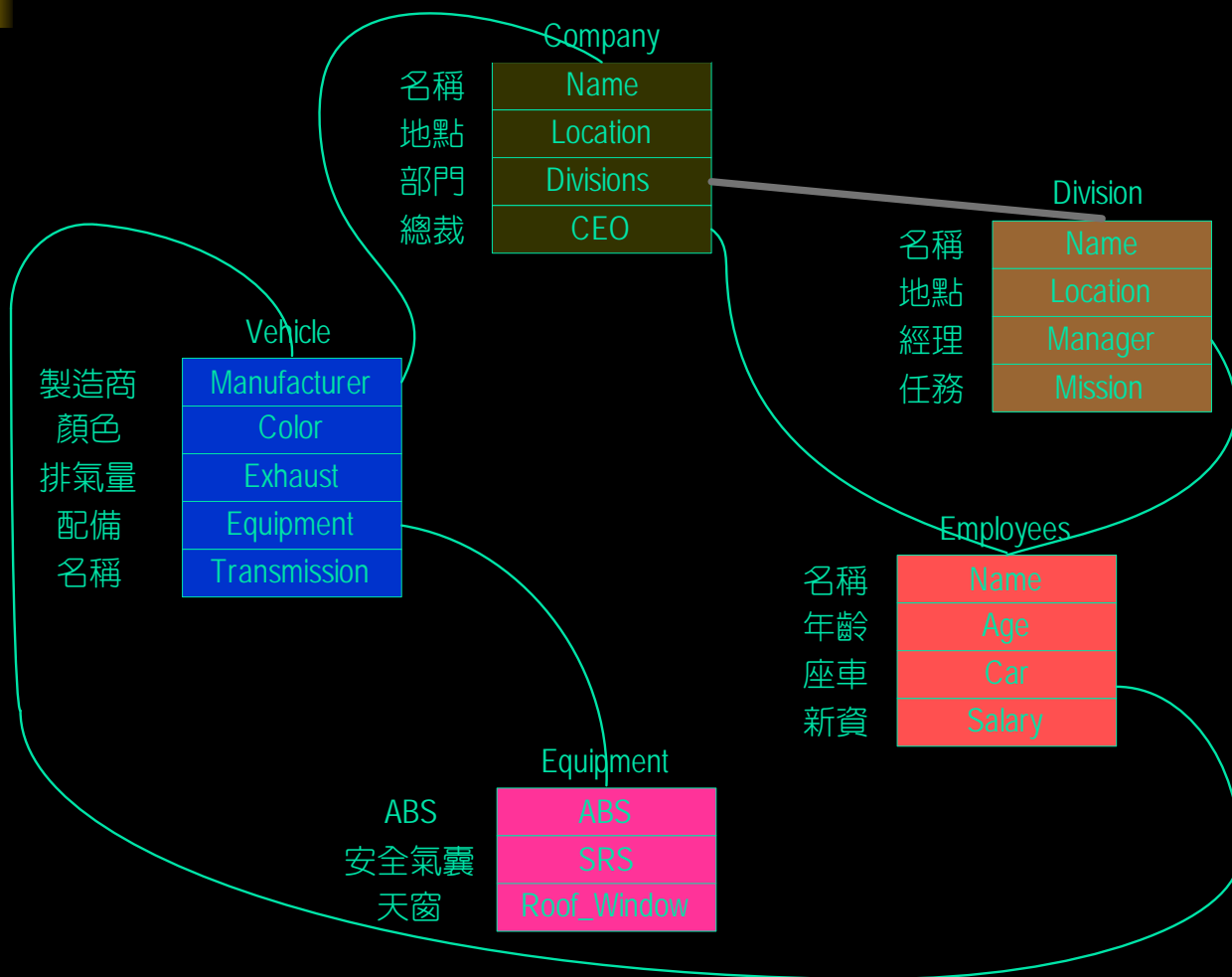
Person							
Order_id	Date	Customer					
		Name	Address				PhoneNo
			ZipCode	State	City	Street	
8	2007/1/26	看成 Customer					
		Frank	看成 Address				7659541
			811	TW	Kaohsiung	Univ. Rd	



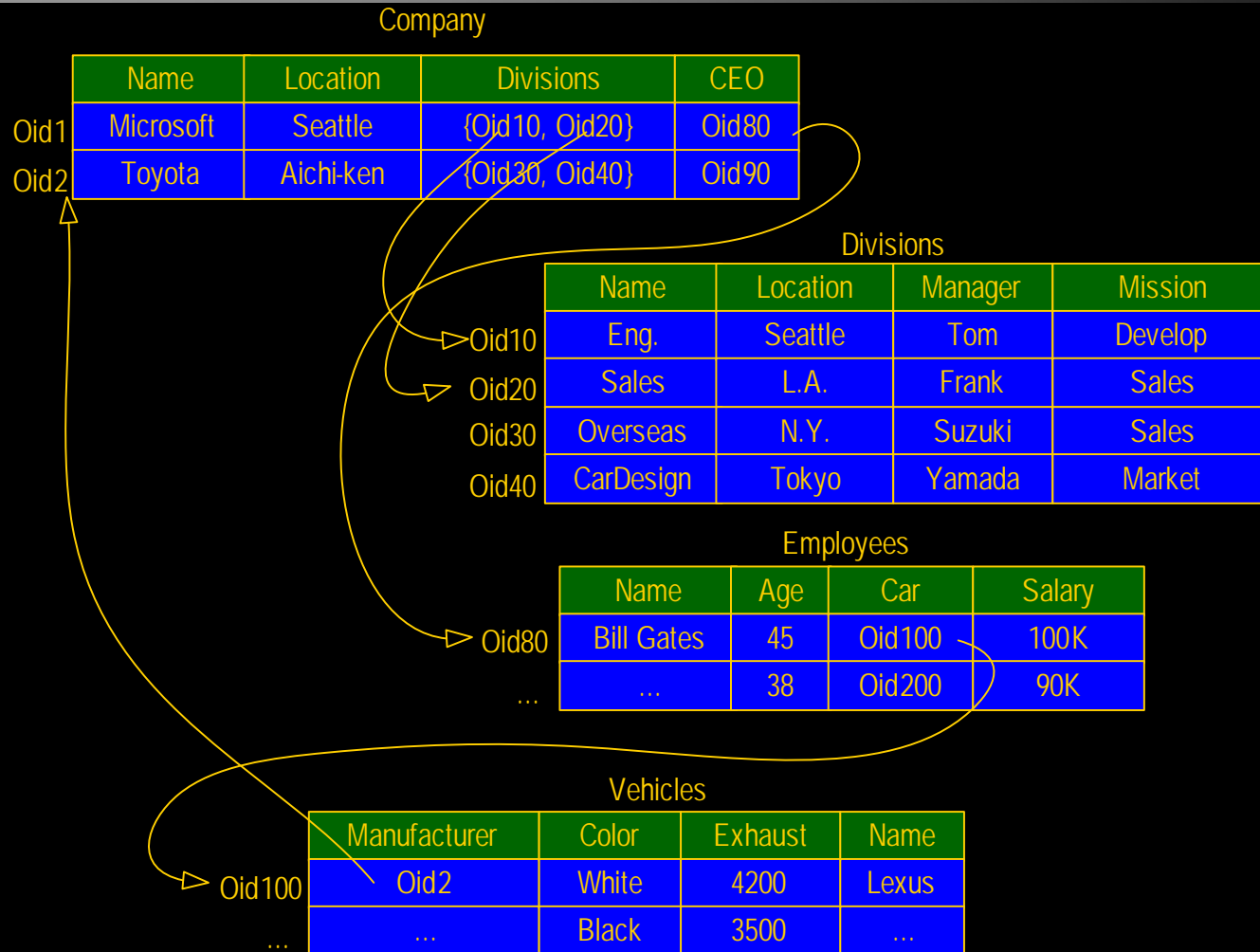
*物件導向/物件關聯式資料模式

- 結合了階層式、網路式與關聯式資料模式後，並且加以擴充而得（但目前並無統一規範，以下僅供參考，可以跳過不討論）
- 由關聯式到物件導向式/物件關聯式歷經了許多的演進與改革（E-R Model, Semantic Data Model, Extended E-R）
- 其資料結構複雜、整合限制條件多，而且其資料上的運算尚未完全標準化（SQL3, OQL）由於模組很多，1998 年起陸續訂出來各個模組

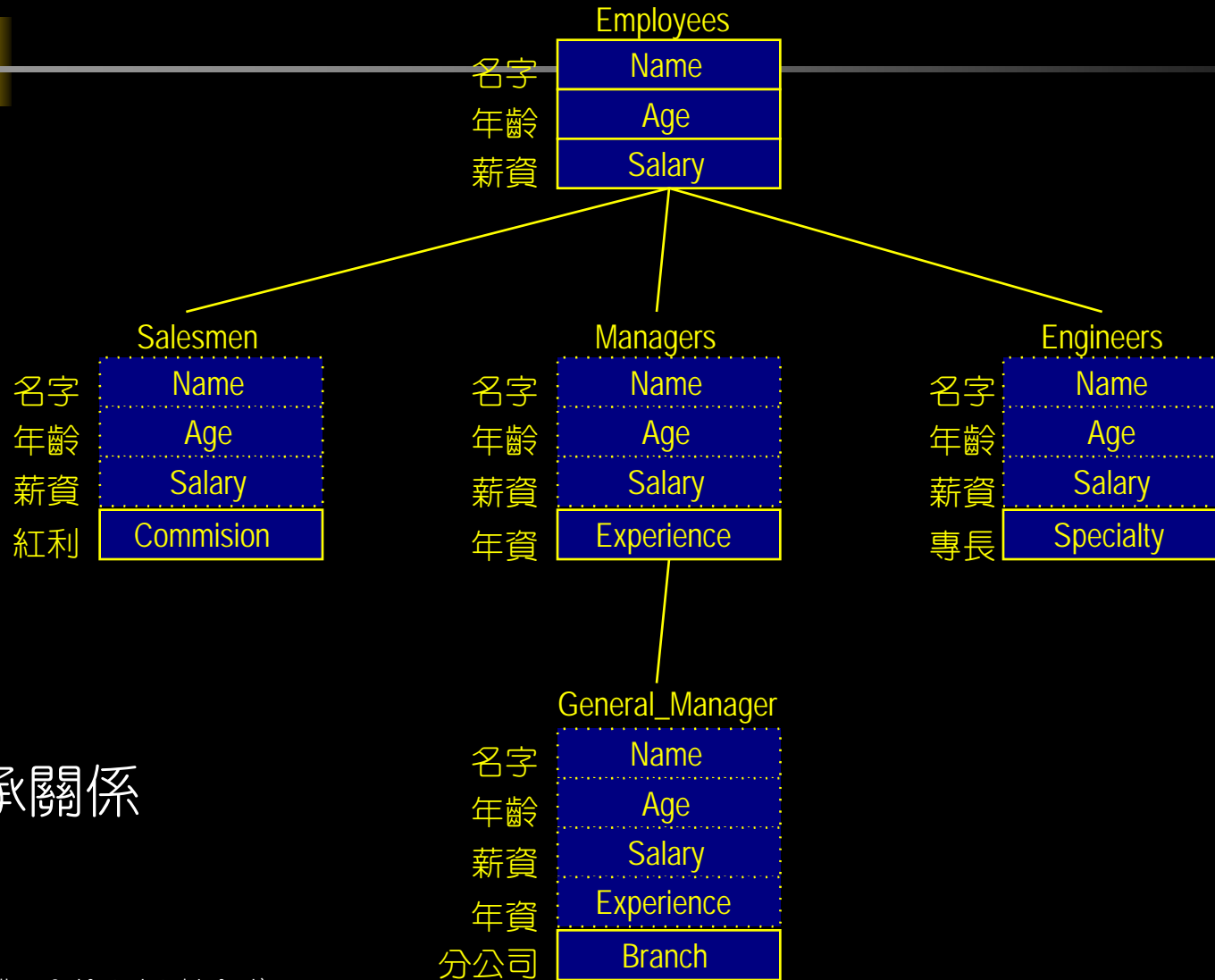
物件導向資料結構



物件導向資料存放



物件導向資料結構 (續)



繼承關係

物件關聯式查詢 (Implicit Join)

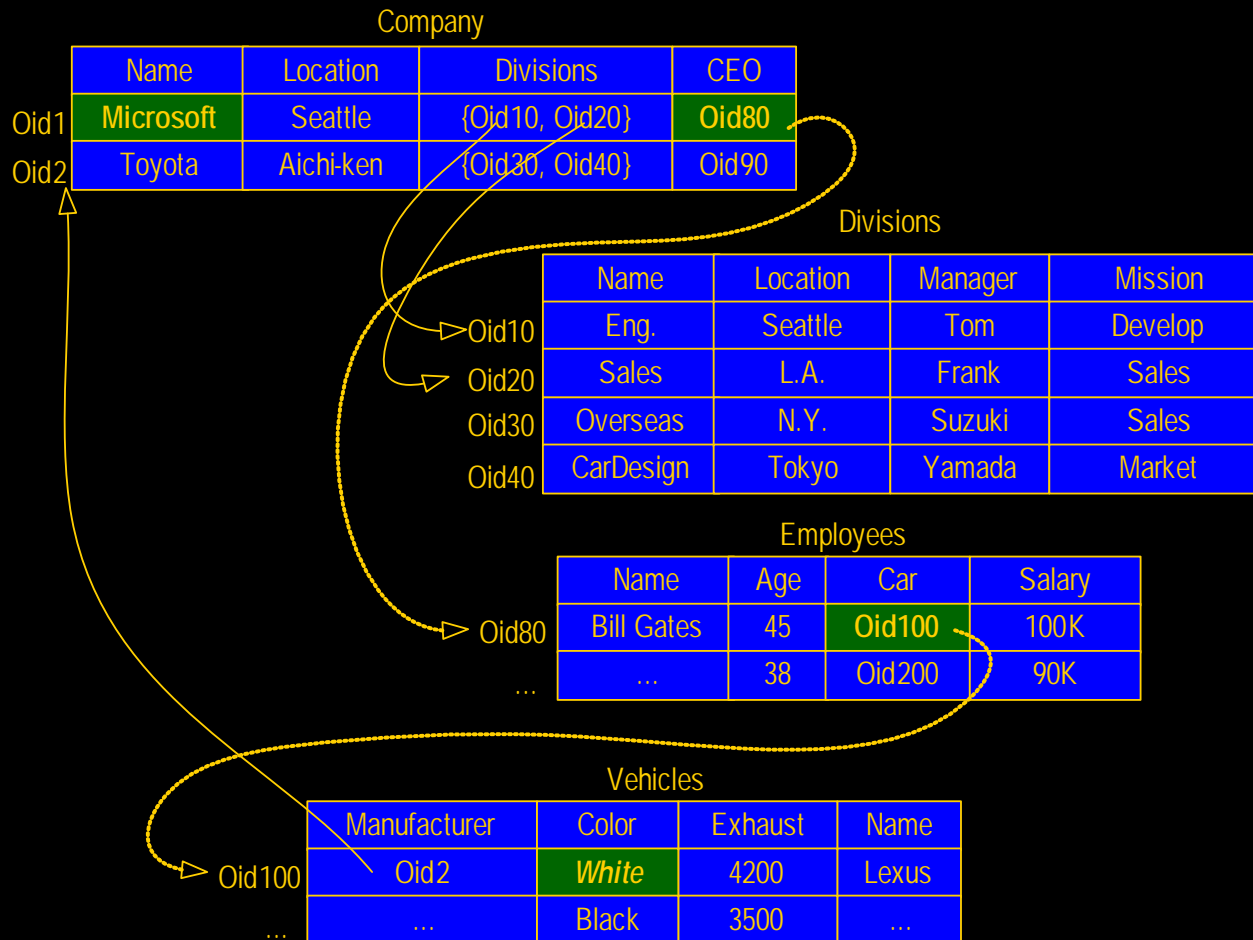
請參考前面的圖

- “Microsoft 這家公司 CEO 的座車顏色為何？”

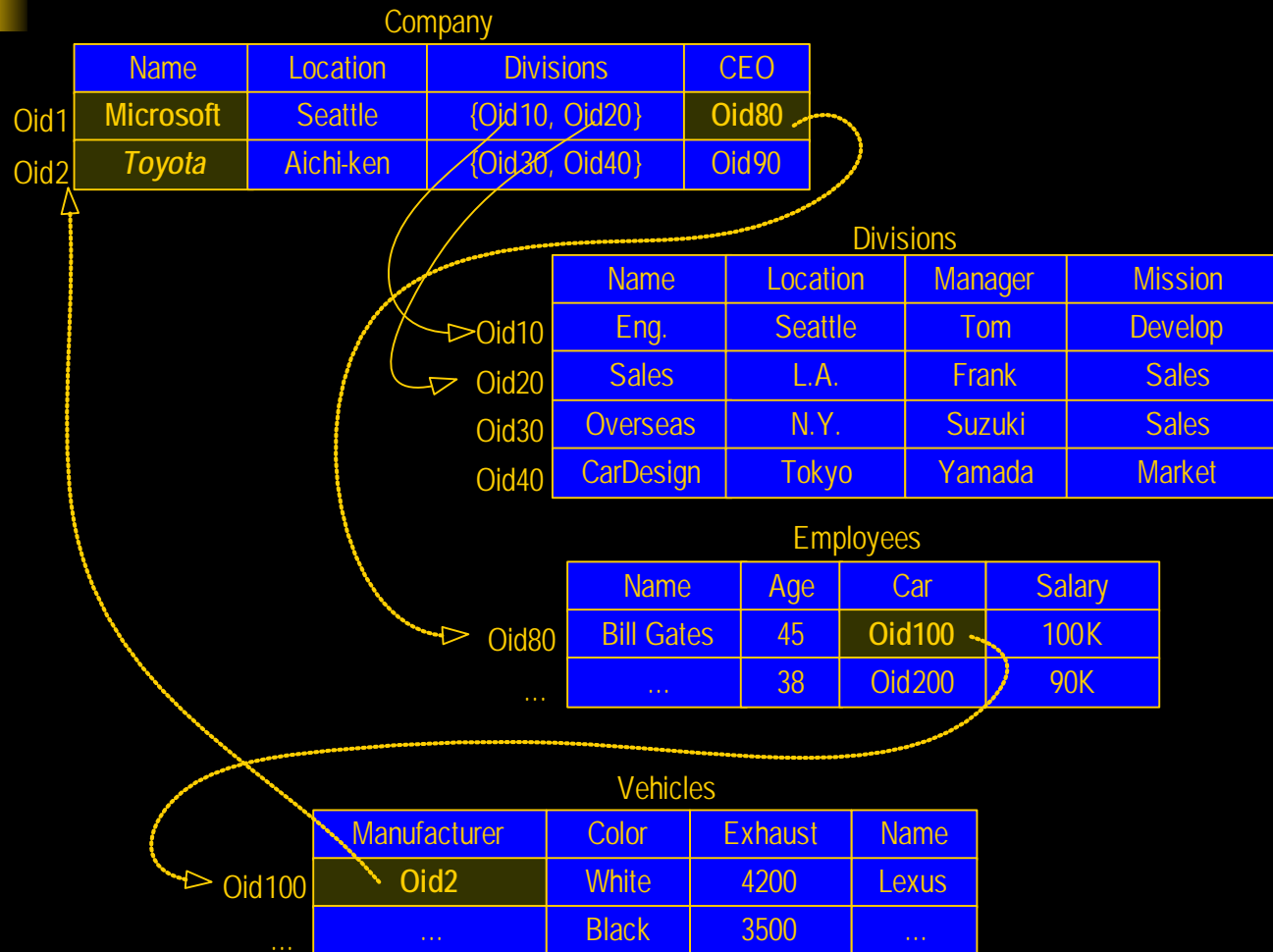
■ `select CEO.Car.Color from Company`
`where Name = 'Microsoft';` (請參考投影片 33, 34 頁的圖)
- “請找出 Microsoft 這家公司 CEO 的座車是由那家公司所生產的？”

■ `select CEO.Car.Manufacturer.Name from Company`
`where Name = 'Microsoft';` (請參考投影片 33, 34 頁的圖)

select CEO.Car.Color from Company
where Name = 'Microsoft' 的資料取得



select CEO.Car.Manufacturer.Name from Company
where Name = 'Microsoft' 的資料取得





*XML 資料模式

- 網際網路的盛行，需要常常在網路上傳送、交換資料，因此 W3C (<http://www.w3c.org>) 推動制定「可擴充式標示語言」(XML, Extensible Markup Language) 標準於 1998 年誕生的。
- XML 是由「國際標準組織」(ISO, International Organization for Standardization) 在1986年所制定的ISO 8879:1986—Standard Generalized Markup Language (SGML) 簡化而來。
- XML 標準讓網際網路應用程式開發工作輕鬆許多



Why not SGML?

- SGML 的目的，是為了制定一個標示語言 (Markup Language) 的標準，讓所有的電子文件都可以使用這套標準從一個應用程式轉換到另一個去。
- 儘管 SGML 的功能很強，可是複雜性卻令人卻步。
- HTML 與 XML 兩者皆是從 SGML 衍生而來的：
 - 雖然 HTML 的語法比較簡單，但其重點在資料的呈現，在網路資料的處理卻有諸多不便；
 - XML 簡化了 SGML 的複雜度，比 HTML 更健全，因而成為重要的網路應用程式開發語言



Why XML? (XML 的特色)

- 具有機器可閱讀的內容資訊：它具有自我描述能力
- 整合傳統資料庫與資料格式。XML 可以描述現存的資料結構，並支援多種資料型別。
- 資料內容與展示方式各自獨立。XML 的設計理念就是希望讓 XML 的資料內容與其顯示方式分開，以保持結構化資料與使用者介面兩者之間的獨立。
資料內容：XML，顯示方式：XSL
- 簡單易懂且能應用於多國語系。XML 支援多語系文件及Unicode。
- 具開放性與延伸性。



XML 在各個領域上的應用

- XML也是其他日漸興起行業的標準，如：健康保健 (HL7)、Wf-XML、RosettaNet、BizTalk、ebXML等，均是目前知名的應用標準。
- 目前所有的關聯式資料庫系統廠商都朝向支援 XML的方向發展。
- 因此，我們覺得「探討XML」也應該漸漸納入資料庫的學習活動中

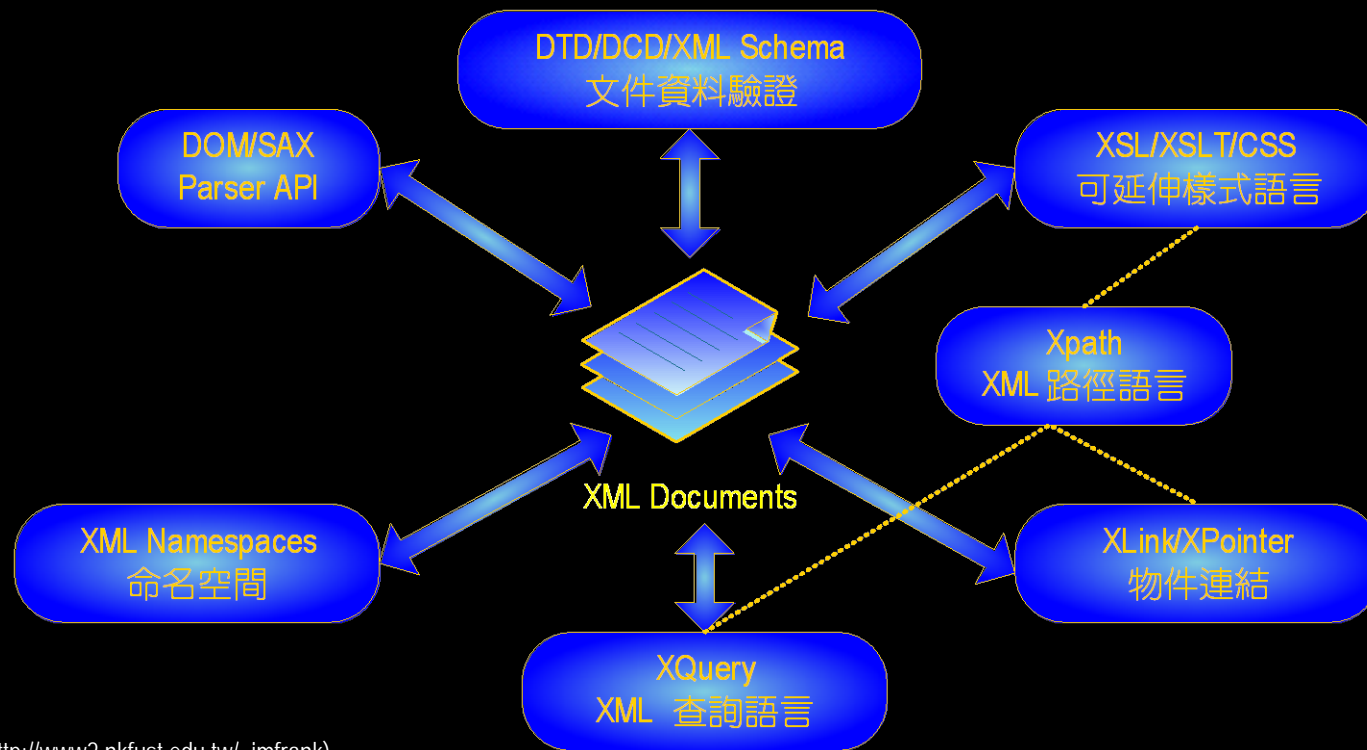


XML vs. Database

- 資料庫系統廠商已陸續支援 XML
 - Oracle 提出了 iFS (Internet File System) 的架構
 - IBM DB2 加上了 XML Extender。
 - SQL Server 2008 也增強了 XML 的功能（後面討論）
 - 直接將資料庫完全 XML 化的資料庫管理系統 (XML Database Management Systems)：像 Software AG 的 Tamino 資料庫系統 (<http://www.softwareag.com>) 以及 Object Design (是eXcelon公司的一個研發部門) 所提出來的 eXcelon (<http://www.exceloncorp.com>) 產品

XML 相關技術

- XML 技術是指一系列根基於 XML 語法所制定之標準與產業規範的集合。





XML 相關技術

- **文件資料驗證 (Document Validation)**：透過 DTD (Document Type Definition)、DCD (Document Content Description) 或 XML Schema 對文件內容做驗證。
- **XML Parser API**：是 XML 文件與應用程式之間的使用介面。目前常用的 XML Parser API 有兩類：Document Object Model (簡稱 DOM) 及 Sample API for XML (簡稱 SAX)。
- **物件連結標準 XLink 與 XPointer**：在 XML 文件中的交互參考超連結，透過 XPointer 達成，XML 文件之間的超連結，則使用 XLink。
- **XQuery 查詢語言** [Chamberlin 2002]：針對 XML 文件所開發出來的查詢語言，也稱為 FLWOR (唸成 Flower) 指令 (因為 XQuery 包含了：For、Let、Where、Order by、Return 五大子句)



XML 相關技術

- XML 命名空間 (XML Naming Space)：主要解決名稱空間衝突的問題，藉著前置一個唯一的命名空間識別項，達到元素及屬性的唯一性。
- 顯示和報表輸出 (Displaying and Reporting)：XML 技術強調將內容與展示方式分開，因此 XML 文件的展示方式可使用
 - CSS (Cascading Style Sheets) 或
 - XSL (Extensible Stylesheet Language) [Royappa 1999] 來定義。
 - XSL 中還包含了 XSLT (XSL Transformation)，讓 XML 文件可以轉換成不同的語法或不同的文件型式。
- 內容定位規則 XPath：提供一個類似於 Unix 檔案路徑定位模式的定位規則 (使用斜線區隔不同元素節點，並以 `@attr_name` 來指定屬性)。XSLT、XLink, XPointer，以及 XQuery 都會用到 XPath。

XML 文件的結構

處理指令 (Processing Instruction)

```
<?xml version="1.0" encoding="Big5" ?>
<?xml-stylesheet type="text/xsl" href="Books.xsl" ?>
```

```
<!--這是兩筆書籍的資料-->
```

註解

```
<書籍資料>
  <書籍>
    <書名>三國演義</書名>
    <作者>羅貫中</作者>
    <價格 幣別="新台幣">120</價格>
  </書籍>
  <書籍>
    <書名>水滸傳</書名>
    <作者>施耐庵</作者>
    <價格 幣別="新台幣">170</價格>
  </書籍>
</書籍資料>
```

根元素



處理指令 (PI, Processing Instruction)

- 起始符號是 '<?' 結束符號是 '?>'，
- 作用是提供此文件的相關訊息給 XML 剖析器 (Parser)。
- 舉例來說，屬於處理指令的「XML宣告」，是要將文件的起始位置告訴 XML 剖析器，所以每一個 XML 文件的第一行都必須是：
`<?xml version= "1.0" ?>`
- 如果要在 XML 文件中使用中文的話，就必須在XML宣告中設定 Big5 中文編碼—encoding= "Big5"
- 若要使用 XSL 排版，就要在文件中加上 XSL 的 PI



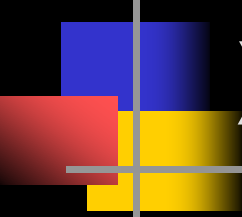
註解 (Comments)

- 註解可以放在文件中的任意位置。
- 註解的起始符號是 '<!--' 結束符號是 '-->'，
- 主要是提供程式設計者說明程式之用，所以 XML 剖析器讀到註解時，不會把它當成指令來檢查。



元素 (Elements)

- 是 XML 文件的主角，在元素裡可以有文字、有子元素，也可以什麼都沒有。
- 我們可以在 XML 文件中重複定義元素 (例如：Books.xml 的「書籍」元素)，但是「根元素」(Root Element) 就只能有一個，而且一定要有一個。
- 所謂「根元素」指的是 XML 文件中最上層的元素，以 Books.xml 來說，其根元素為「書籍資料」



XML 屬性與本文

- **屬性 (Attribute)**：XML 元素除了標籤 (Tag) 以外，還可以定義屬性 (Attribute)，用來進一步描述更多資訊，屬性可以定義多個，而且彼此間沒有順序之分。
- **本文 (Text)**：本文是用來存放元素資料的實質內容。



XML 的基本語法

- XML 文件是由標籤 (Tag) 與本文 (Text) 所構成的。
- 標籤是以角括號 (<>) 夾起來，主要是用來將資料定義成一個元素 (Element)，而本文則是提供資料的實質內容。
- 定義一個 XML 元素的基本語法如下：
<name-of-element attribute1 = "xxx" ...>
text content
</name-of-element>
例如: <價格 貨幣單位= "美金" >500</價格>
- 如果元素裡沒有資料，那麼就稱作「空元素」 (Empty Element)。
其寫法可簡化為 *<name-of-element />*



XML 的基本語法

- 為了能夠清楚的描述資料的結構，XML文件允許使用者可以在元素裡定義「子元素」，形成巢狀 (Nested) 結構：

```
<書籍>  
  <書名>三國演義</書名>  
  <作者>羅貫中</作者>  
  <價格>120</價格>  
</書籍>
```



XML的整合限制條件

- 對 XML 剖析器來說，能夠檢驗過關的 XML 文件，有以下兩大類：
 - 符合正確格式的 XML 文件 (Well-Formed XML Documents)
 - 合法的 XML 文件 (Valid XML Documents): 需要配合 DTD (Document Type Definition) 來驗證內容的正確性



Well-Formed XML Documents

- 第一行一定是 XML 宣告。
- 元素的起始標籤與結束標籤必須成對出現。也可以以 “空元素” 方式出現。
- 一份 XML 文件只能有一個根元素。
- 所有標籤必須滿足巢狀的結構，同一層級的不同標籤彼此之間不可以交錯出現：
- 使用英文字母作為標籤名稱時，大小寫有區分
- 屬性值必須以 “” 或 “ ” 夾起來。
- 有五個特殊字元（分別是 <, >, ", ', &）不能直接用在元素的資料上，若要使用必須使用其替代符號



XML 文件中五個特殊字元的表示法

特殊字元	替代符號	代 表 意 義
<	<	<u>l</u> ower <u>t</u> han
>	>	<u>g</u> reater <u>t</u> han
"	"	<u>q</u> uotation mark
'	'	<u>a</u> postrophe
&	&	<u>a</u> mpersand



Valid XML Documents

- 仰賴所謂的 DTD (Document Type Definition) 來檢查文件內容的合法性，
- 該 XML 文件如果完全遵循 DTD 或 XML Schema 所制定的規則來安排文件結構的話，那麼我們就稱該文件為合法的 (Valid) 的 XML 文件。
- 由於 DTD 無法規範文件內容的資料型態，而且缺乏擴充性，所以業界又制定了 XML Schema 提供另一選擇。
- 有興趣的讀者請參閱相關書籍，例如：由 E.R. Harold 所撰寫的 XML Bible [E.R. Harold (1999)]



XML 文件的型式

- 以資料為主的 XML 文件 (**Data-Centric XML Document**) : 主要適用於機器可以自動判讀的結構化資料，如：訂單、課表、或價目表、班機時刻表等。本章中所有的 XML 文件範例都屬於此類
- 以文件為主的 XML 文件 (**Document-Centric XML Document**) : 主要適用於半結構化 (Semi-Structured) 或非結構化 (Non-Structured) 的不規則文件資料，例如：由網站內容所存成的 XML 文件，即屬於此類。



資料在 XML 文件的寫法分類

- 以屬性為主 (Attribute-Centric)：也就是將大部分的資料放在元素的屬性值上。例如，以下的 XML 文件中，資料“三國演義”是放在「書籍」這個空元素的「書名」屬性值上。

```
<?xml version="1.0" encoding="Big5" ?>
<訂單>
  <書籍 書名="三國演義" 作者="羅貫中" 價格="120" />
  <書籍 書名="水滸傳" 作者="施耐庵" 價格="170" />
  <書籍 書名="西遊記" 作者="吳承恩" 價格="140" />
</訂單>
```



資料在 XML 文件的寫法分類

以元素為主 (Element-Centric)：也就是依資料的性質來定義的元素，並將資料放在元素內，不以屬性的型式來表示。如以下 XML 文件中，資料“三國演義”是放在「書名」元素內。

```
<?xml version="1.0" encoding="Big5" ?>
<訂單>
<書籍><書名>三國演義</書名><作者>羅貫中</作者><價格>120</價格></書籍>
<書籍><書名>水滸傳</書名><作者>施耐庵</作者><價格>170</價格></書籍>
<書籍><書名>西遊記</書名><作者>吳承恩</作者><價格>140</價格></書籍>
</訂單>
```



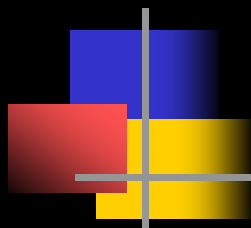
Attribute- vs. Element-Centric

- 以屬性為主的 XML 標示型式 (Attribute-Centric XML Forms) 具有下列優勢：
 - 同一個元素中的屬性之間沒有順序之分，可以任意擺放，
 - 可以加上資料型態 (Data Type) 的描述屬性，讓資料更容易處理，
 - 可以限定容許出現的資料值為何，
 - 整個文件的內容較小，比較節省儲存空間。
- 以元素為主的 XML 文件 (Element-Centric XML Document) 則具有容易做轉換的優勢。



XML 的資料運算

- XML 資料的運算一般是透過其專屬的方法 (Methods)，配合 **XQuery** 與 **XPath** 的語法來進行運算。
- 在SQL Server 2008中，這些專屬的方法有：
 - `exist(XQuery)`、
 - `value('XQuery', 'SQL Type')`、
 - `nodes (XQuery) as Table(Column)`，以及
 - `modify(XML_DML)` 等方法，我們在本書 7.10.5 節有簡單的說明與範例



本章結束
The End.