

# **Transact-SQL 語法教學**

**在 SQL Server 使用 DML 資料處理語言**

# 課程大綱

- 關聯式資料庫邏輯結構
- SQL Server Management Studio 操作說明
- SELECT 基本句型
- 資料彙總統計
- 以 JOIN 結合多個資料表
- 活用于查詢 ( Subquery )
- 以 Insert 、 Update 、 Delete 修改資料
- 善用索引加快查詢效能

# 關聯式資料庫邏輯結構

- 資料以橫列直欄的方式組織於二維表格（ **Table** ）之中，各資料表（ **Table** ）存放現實世界中的實體或概念上認定存在的東西，例如：學生資料表、班級資料表、員工資料表。
- 每一直欄稱為欄位（ **Field** ）。
- 每一橫列稱為記錄（ **Record** ）。
- 每個資料表都各有其主鍵（ **Primary Key** ， **PK** ）。
- 必要時，以某個欄位為外鍵（ **Foreign Key** ， **FK** ）**關聯**到另一資料表的主鍵以獲得進一步的相關資料。

# 關聯式資料庫邏輯結構

- 每一直欄稱為欄位 ( Field ) 。

<i>CityID</i>	<i>CityName</i>
TP	台北
TC	台中
KS	高雄

<i>EmpID</i>	<i>LastName</i>	<i>FirstName</i>	<i>CtryID</i>	<i>Extension</i>	<i>LastMod</i>
integer	longstring	varchar(20)	char(2)	char(6)	longstring
101	Wang	Angle	TP	x19891	\HR\KarlID
102	Chien	Wolfgang	TC	x19433	\HR\KarlID
103	Martin	Jose	TP	x21467	\HR\AmyL

# 關聯式資料庫邏輯結構

- 每一橫列稱為**記錄 ( Record )**。

<i>CityID</i>	<i>CityName</i>
TP	台北
TC	台中
KS	高雄

<i>EmpID</i>	<i>LastName</i>	<i>FirstName</i>	<i>CtryID</i>	<i>Extension</i>	<i>LastMod</i>
integer	longstring	varchar(20)	char(2)	char(6)	longstring
101	Wang	Angle	TP	x19891	\HR\KarlID
102	Chien	Wolfgang	TC	x19433	\HR\KarlID
103	Martin	Jose	TP	x21467	\HR\AmyL

# 關聯式資料庫邏輯結構

- 每個資料表都各有其主鍵 ( **Primary Key** , **PK** ) 。
- 必要時，以某個欄位為外鍵 ( **Foreign Key** , **FK** ) 關聯到另一資料表的主鍵以獲得進一步的相關資料。

<b>PK</b>	
<i>CityID</i>	<i>CityName</i>
TP	台北
TC	台中
KS	高雄

<b>PK</b>		<b>FK</b>			
<i>EmpID</i>	<i>LastName</i>	<i>FirstName</i>	<i>CtryID</i>	<i>Extension</i>	<i>LastMod</i>
integer	longstring	varchar(20)	char(2)	char(6)	longstring
101	Wang	Angle	TP	x19891	\HR\KarlID
102	Chien	Wolfgang	TC	x19433	\HR\KarlID
103	Martin	Jose	TP	x21467	\HR\AmyL

# **SELECT 基本句型**

# 利用 **SELECT** 查詢資料

- **SELECT** 敘述基本語法
- 指定欄位清單
- 資料排序
- 篩選資料
  - WHERE 子句句型 – 比較型
  - WHERE 子句句型 – 樣式比對型
  - WHERE 子句句型 – 區間型
  - WHERE 子句句型 – 列舉型
- 格式化結果集



# SELECT 敘述基本語法

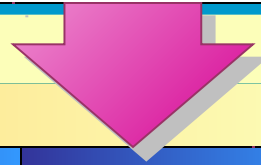
- **<select\_list>**: 以逗號條列各個欄位
- 在 **FROM** 子句指定資料表名稱
- 以 **WHERE** 指定篩選欄位
- 以 **ORDER BY** 指定排序欄位

## 基本語法

```
SELECT [ALL | DISTINCT] <select_list>  
FROM {<table_source>} [,...n]  
WHERE <search_condition>  
ORDER BY <field name> [ASC | DESC] [,...n]
```

# 指定欄位清單

```
USE northwind
SELECT employeeid, lastname, firstname, title
FROM employees
GO
```



<i>employeeid</i>	<i>lastname</i>	<i>firstname</i>	<i>title</i>
1	Davolio	Nancy	Sales Representative
2	Fuller	Andrew	Vice President, Sales
3	Leverling	Janet	Sales Representative
4	Peacock	Margaret	Sales Representative
5	Buchanan	Steven	Sales Manager
6	Suyama	Michael	Sales Representative
7	King	Robert	Sales Representative
8	Callahan	Laura	Inside Sales Coordinator
9	Dodsworth	Anne	Sales Representative

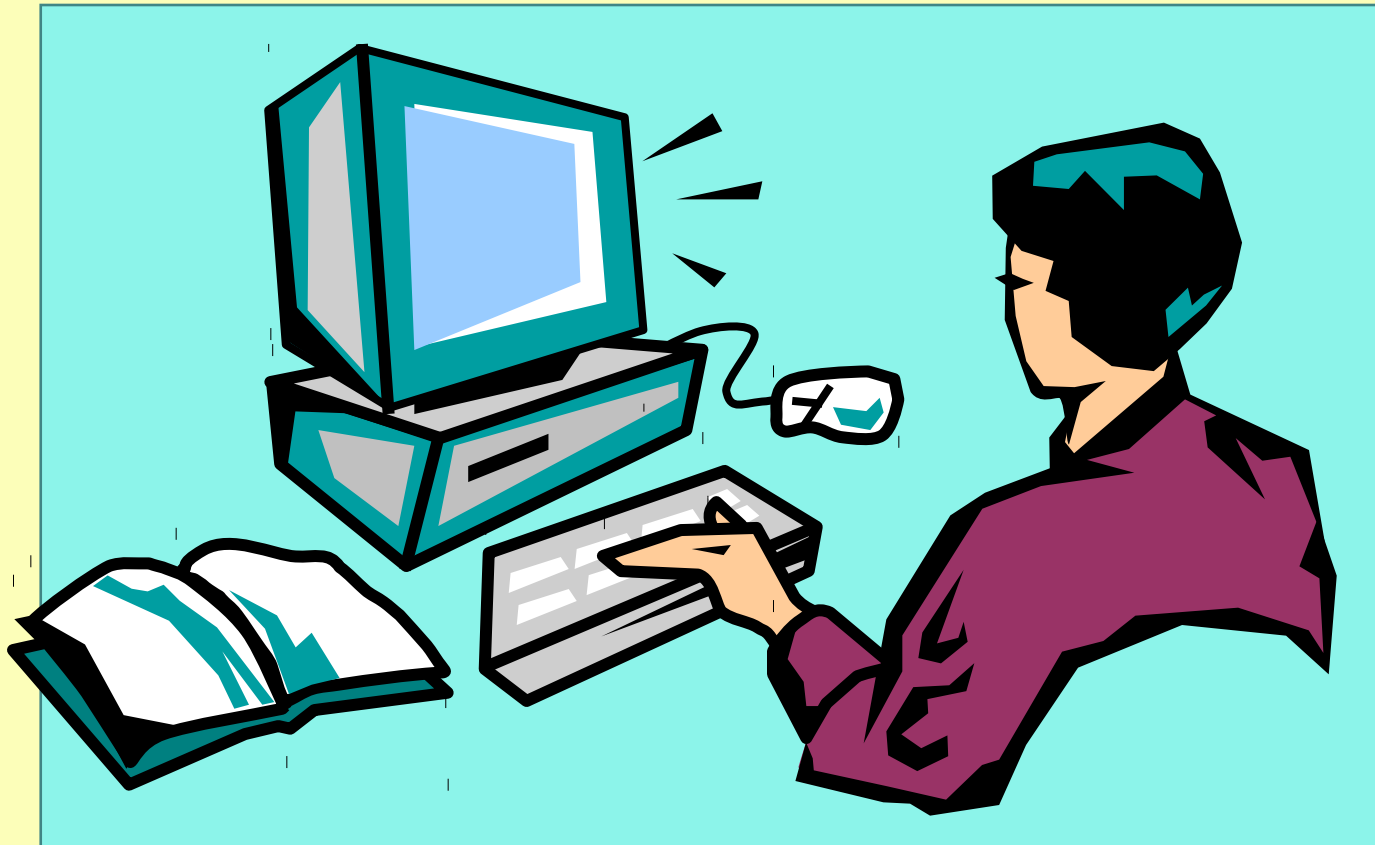
# 資料排序

```
USE northwind
SELECT productid, productname, categoryid, unitprice
FROM products
ORDER BY categoryid, unitprice DESC
GO
```



<i>productid</i>	<i>productname</i>	<i>categoryid</i>	<i>unitprice</i>
38	Cote de Blaye	1	263.5000
43	Ipoh Coffee	1	46.0000
2	Chang	1	19.0000
...	...	...	...
63	Vegie-spread	2	43.9000
8	Northwoods Cranberry Sauce	2	40.0000
61	Sirop d'érable	2	28.5000
...	...	...	...

# 練習：基本 SELECT 句型



## ◆ 篩選資料

- **WHERE 子句句型 – 比較型**
- **WHERE 子句句型 – 樣式比對型**
- **WHERE 子句句型 – 區間型**
- **WHERE 子句句型 – 列舉型**

# WHERE 子句句型 – 比較型 (一)

```
USE northwind
SELECT employeeid, lastname, firstname, title
FROM employees
WHERE employeeid = 5
GO
```

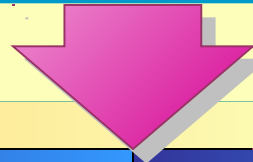


<i><b>employeeid</b></i>	<i><b>lastname</b></i>	<i><b>firstname</b></i>	<i><b>title</b></i>
5	Buchanan	Steven	Sales Manager

# WHERE 子句句型 – 比較型 (二)

## Example 1

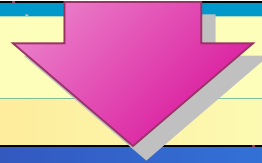
```
USE northwind  
SELECT lastname, city  
FROM employees  
WHERE country = 'USA'  
GO
```



<i>lastname</i>	<i>city</i>
Davolio	Seattle
Fuller	Tacoma
Leverling	Kirkland
Peacock	Redmond
Callahan	Seattle

# WHERE 子句句型 – 樣式比對型

```
USE northwind  
SELECT companyname  
FROM customers  
WHERE companyname LIKE '%Restaurant%'  
GO
```



<i><b>companyname</b></i>
GROSELLA-Restaurante
Lonesome Pine Restaurant
Tortuga Restaurante



# 運用邏輯運算元

```
USE northwind
SELECT productid, productname, supplierid, unitprice
FROM products
WHERE (productname LIKE 'T%' OR productid = 46)
AND (unitprice > 16.00)
GO
```



<i><b>productid</b></i>	<i><b>productname</b></i>	<i><b>supplierid</b></i>	<i><b>unitprice</b></i>
14	Tofu	6	23.25
29	Thüringer Rostbratwurst	12	123.79
62	Tarte au sucre	29	49.3

# WHERE 子句句型 – 區間型

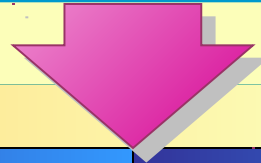
```
USE northwind
SELECT productname, unitprice
FROM products
WHERE unitprice BETWEEN 10 AND 20
GO
```



<i>productname</i>	<i>unitprice</i>
Chai	18
Chang	19
Aniseed Syrup	10
Genen Shouyu	15.5
Pavlova	17.45
Sir Rodney's Scones	10
...	...

# WHERE 子句句型 – 列舉型

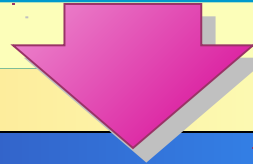
```
USE northwind
SELECT companyname, country
FROM suppliers
WHERE country IN ('Japan', 'Italy')
GO
```



<i><b>companyname</b></i>	<i><b>country</b></i>
Tokyo Traders	Japan
Mayumi's	Japan
Formaggi Fortini s.r.l.	Italy
Pasta Buttini s.r.l.	Italy

# NULL = Unknown 「未知」

```
USE northwind
SELECT companyname, fax
FROM suppliers
WHERE fax IS NULL
GO
```




<i><b>companyname</b></i>	<i><b>fax</b></i>
Exotic Liquids	NULL
New Orleans Cajun Delights	NULL
Tokyo Traders	NULL
Cooperativa de Quesos 'Las Cabras'	NULL
...	...

## ◆ 格式化結果集

- 排序資料
- **DISTINCT**
- 變更欄位名稱
- 利用靜態文字修飾資料
- **RANK()** 進階排序函數

# 排序資料

```
USE northwind
SELECT productid, productname, categoryid, unitprice
FROM products
ORDER BY categoryid, unitprice DESC
GO
```



<i><b>productid</b></i>	<i><b>productname</b></i>	<i><b>categoryid</b></i>	<i><b>unitprice</b></i>
38	Cote de Blaye	1	263.5000
43	Ipoh Coffee	1	46.0000
2	Chang	1	19.0000
...	...	...	...
63	Vegie-spread	2	43.9000
8	Northwoods Cranberry Sauce	2	40.0000
61	Sirop d'érable	2	28.5000
...	...	...	...

# DISTINCT ， 若記錄內容完全相同， 只留一筆

```
USE northwind
SELECT DISTINCT country
FROM suppliers
ORDER BY country
GO
```



<i>country</i>
Australia
Brazil
Canada
Denmark
Finland
France
Germany
Italy
Japan
Netherlands
Norway
Singapore
Spain
Sweden
UK
USA

# 變更欄位名稱

```
USE northwind
SELECT  firstname AS First, lastname AS Last
        ,employeeid AS 'Employee ID:'
FROM employees
GO
```



<i><b>First</b></i>	<i><b>Last</b></i>	<i><b>Employee ID:</b></i>
Nancy	Davolio	1
Andrew	Fuller	2
Janet	Leverling	3
Margaret	Peacock	4
Steven	Buchanan	5
Michael	Suyama	6
Robert	King	7
Laura	Callahan	8
Anne	Dodsworth	9



# 利用靜態文字修飾資料

```
USE northwind
SELECT  Firstname, Lastname,
        'Identification number:' +
        Convert(varchar(2), employeeid) as ID
FROM employees
GO
```

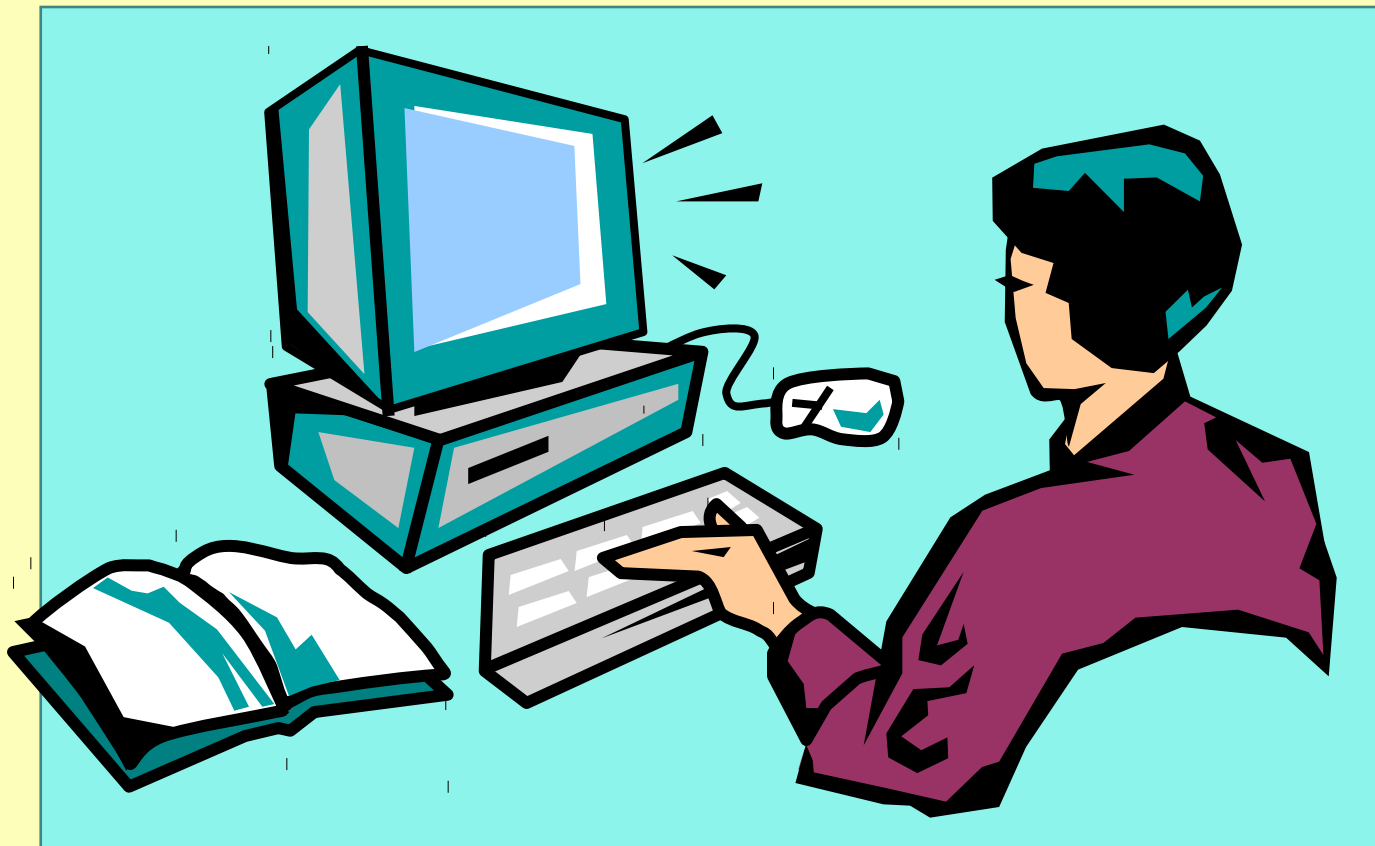


<i><b>Firstname</b></i>	<i><b>Lastname</b></i>	<i><b>ID</b></i>
Nancy	Davolio	Identification Number: 1
Andrew	Fuller	Identification Number: 2
Janet	Leverling	Identification Number: 3
Margaret	Peacock	Identification Number: 4
Steven	Buchanan	Identification Number: 5
Michael	Suyama	Identification Number: 6
Robert	King	Identification Number: 7
Laura	Callahan	Identification Number: 8
Anne	Dodsworth	Identification Number: 9

# 進階排序函數

函數	功能描述
<b>RANK</b>	傳回該筆記錄的排序編號（可能因為相同欄位值而有跳號的情形）
<b>DENSE_RANK</b>	傳回該筆記錄的排序編號（一定會連號）
<b>ROW_NUMBER</b>	傳回該筆記錄的位置編號
<b>NTILE</b>	將資料分成 N 組（層）後，傳回該筆記錄所屬的分組編號。

# 練習：各項資料篩選句型與資料格式化



# 資料彙總統計

# 資料彙總統計

- **TOP N** ，列出排頭的資料
- 使用彙總函數
- 彙總函數與 Null 值
- 使用 **GROUP BY** 子句
  - 以 **HAVING** 進行彙總之後的第二階段條件過濾
  - 以 **GROUP BY** 配合 **ROLLUP** 進行多層次統計
- **COMPUTE** 與 **COMPUTE BY** 子句

# TOP N ，列出排頭的資料

- 僅傳回結果集的前 N 筆記錄
- 排列次序由 **ORDER BY** 子句決定
- 若加註 **WITH TIES** ，增額加列邊界值相同的記錄

## 範例 1

```
USE northwind
SELECT TOP 5 orderid, productid, quantity
FROM [order details]
ORDER BY quantity DESC
GO
```

## 範例 2

```
USE northwind
SELECT TOP 5 WITH TIES orderid, productid, quantity
FROM [order details]
ORDER BY quantity DESC
GO
```

## ◆使用彙總函數

函數名稱	功能描述
AVG	計算平均值
COUNT	有資料的共有幾筆
COUNT (*)	一共多少筆 ( 有 Null 值的記錄也算進去 )
MAX	傳回最大值
MIN	傳回最小值
SUM	計算總和
STDEV	計算標準差
VAR	計算變異數

# 彙總函數與 Null 值

- 絕大多數的彙總函數均排除 Null，不列入計算
- **COUNT(\*)** 例外，有 Null 值的資料仍然計入一筆

## 範例 1

```
USE northwind
SELECT COUNT (*)
  FROM employees
GO
```

## 範例 2

```
USE northwind
SELECT COUNT(reportsto)
  FROM employees
GO
```




# 使用 GROUP BY 子句


```
USE northwind
SELECT productid, orderid, quantity
FROM orderhist
GO
```

<i>productid</i>	<i>orderid</i>	<i>quantity</i>
1	1	5
1	1	10
2	1	10
2	2	25
3	1	15
3	2	30

```
USE northwind
SELECT productid, SUM(quantity) AS total_quantity
FROM orderhist
GROUP BY productid
GO
```



<i>productid</i>	<i>total_quantity</i>
1	15
2	35
3	45



<i>productid</i>	<i>total_quantity</i>
2	35


```
USE northwind
SELECT productid, SUM(quantity) AS total_quantity
FROM orderhist
WHERE productid = 2
GROUP BY productid
GO
```

# 以 HAVING 進行彙總之後的第二階段條件過濾

```
USE northwind
SELECT productid,orderid,quantity
FROM orderhist
GO
```

<i>productid</i>	<i>orderid</i>	<i>quantity</i>
1	1	5
1	1	10
2	1	10
2	2	25
3	1	15
3	2	30

```
USE northwind
SELECT productid, SUM(quantity)
      AS total_quantity
FROM orderhist
GROUP BY productid
HAVING SUM(quantity)>=30
GO
```



<i>productid</i>	<i>total_quantity</i>
2	35
3	45

# 以 GROUP BY 配合 ROLLUP 進行多層次統計

```
USE northwind
SELECT productid,orderid,SUM(quantity) AS total_quantity
FROM orderhist
GROUP BY productid,orderid
WITH ROLLUP
ORDER BY productid,orderid
GO
```

<i>productid</i>	<i>orderid</i>	<i>total_quantity</i>
NULL	NULL	95
1	NULL	15
1	1	5
1	2	10
2	NULL	35
2	1	10
2	2	25
3	NULL	45
3	1	15
3	2	30

# COMPUTE 與 COMPUTE BY 子句

## COMPUTE

```
USE northwind
SELECT productid, orderid
       , quantity
FROM orderhist
ORDER BY productid, orderid
COMPUTE SUM(quantity)
GO
```

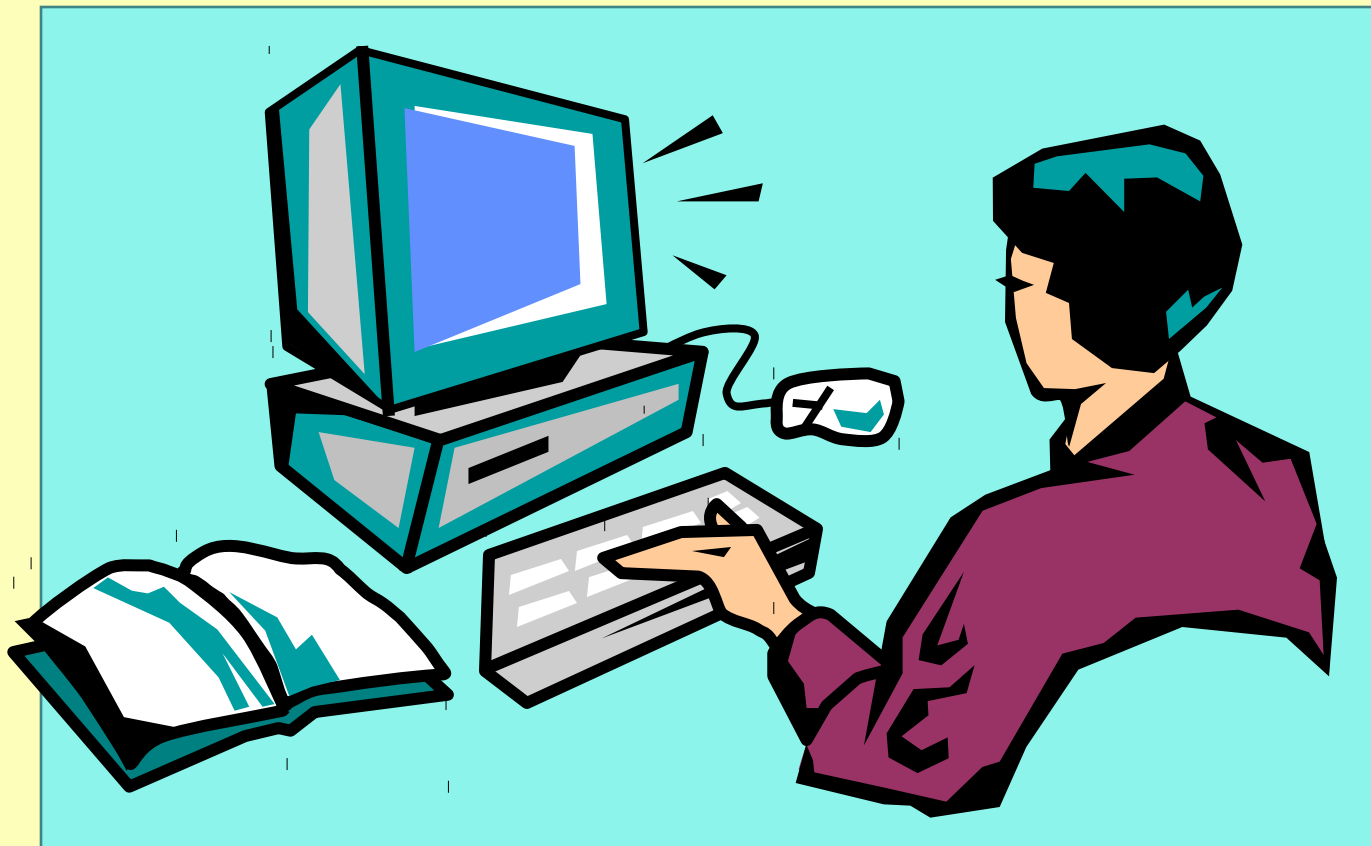
<i>productid</i>	<i>orderid</i>	<i>quantity</i>
1	1	5
1	2	10
2	1	10
2	2	25
3	1	15
3	2	30
sum		95

## COMPUTE BY

```
USE northwind
SELECT productid, orderid, quantity
FROM orderhist
ORDER BY productid, orderid
COMPUTE SUM(quantity) BY productid
COMPUTE SUM(quantity)
GO
```

<i>productid</i>	<i>orderid</i>	<i>quantity</i>
1	1	5
1	2	10
sum		15
2	1	10
2	2	25
sum		35
3	1	15
3	2	30
sum		45
sum		95

# 練習：資料彙總統計



# JOIN

## 結合多個資料表

# ◆ JOIN 結合多個資料表

- 資料表別名
- Join 語法說明
  - Inner Join
  - Outer Join
  - Cross Join
  - Self Joining
- 以 **UNION** 上下整併資料表

# 資料表別名

## ■ 範例 1 ( 未使用資料表別名 )

```
USE joindb
SELECT buyer_name, sales.buyer_id, qty
FROM buyers INNER JOIN sales
ON buyers.buyer_id = sales.buyer_id
GO
```

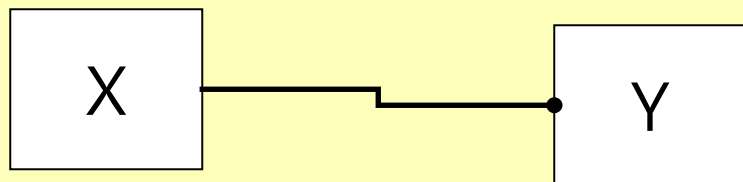
## ■ 範例 2 ( 使用資料表別名 )

```
USE joindb
SELECT buyer_name, s.buyer_id, qty
FROM buyers AS b INNER JOIN sales AS s
ON b.buyer_id = s.buyer_id
GO
```



# Join 語法說明

- 以 **JOIN** 註明另一個資料表
- 再以 **ON** 指定結合條件
- 通常利用主鍵與外鍵欄位指定 **ON** 條件
- 欄位名稱重複時，須加註資料表名稱（或別名）



```
SELECT X. 欄位 , Y. 欄位 , ...  
FROM X JOIN Y ON Y.FK = X.PK
```

# Inner Join

```
USE joindb
SELECT buyer_name, sales.buyer_id, qty
FROM buyers INNER JOIN sales
ON buyers.buyer_id = sales.buyer_id
GO
```

**buyers**

<i>buyer_name</i>	<i>buyer_id</i>
Adam Barr	1
Sean Chai	2
Eva Corets	3
Erin O'Melia	4

**sales**

<i>buyer_id</i>	<i>prod_id</i>	<i>qty</i>
1	2	15
1	3	5
4	1	37
3	5	11
4	2	1003

**Result**

<i>buyer_name</i>	<i>buyer_id</i>	<i>qty</i>
Adam Barr	1	15
Adam Barr	1	5
Erin O'Melia	4	37
Eva Corets	3	11
Erin O'Melia	4	1003

# Outer Join

```
USE joindb
SELECT buyer_name, sales.buyer_id, qty
FROM buyers LEFT OUTER JOIN sales
ON buyers.buyer_id = sales.buyer_id
GO
```

**buyers**

<i>buyer_name</i>	<i>buyer_id</i>
Adam Barr	1
Sean Chai	2
Eva Corets	3
Erin O'Melia	4

**sales**

<i>buyer_id</i>	<i>prod_id</i>	<i>qty</i>
1	2	15
1	3	5
4	1	37
3	5	11
4	2	1003

**Result**

<i>buyer_name</i>	<i>buyer_id</i>	<i>qty</i>
Adam Barr	1	15
Adam Barr	1	5
Erin O'Melia	4	37
Eva Corets	3	11
Erin O'Melia	4	1003
Sean Chai	NULL	NULL

# Cross Join

```
USE joindb
SELECT buyer_name, qty
FROM buyers
CROSS JOIN sales
GO
```

**buyers**

<i>buyer_id</i>	<i>buyer_name</i>
1	Adam Barr
2	Sean Chai
3	Eva Corets
4	Erin O'Melia

**sales**

<i>buyer_id</i>	<i>prod_id</i>	<i>qty</i>
1	2	15
1	3	5
4	1	37
3	5	11
4	2	1003

**Result**

<i>buyer_name</i>	<i>qty</i>
Adam Barr	15
Adam Barr	5
Adam Barr	37
Adam Barr	11
Adam Barr	1003
Sean Chai	15
Sean Chai	5
Sean Chai	37
Sean Chai	11
Sean Chai	1003
Eva Corets	15
...	...

# 結合更多的資料表

```
SELECT buyer_name, prod_name, qty
FROM buyers
INNER JOIN sales
  ON buyers.buyer_id = sales.buyer_id
INNER JOIN produce
  ON sales.prod_id = produce.prod_id
GO
```

**buyers**

<i>buyer_id</i>	<i>buyer_name</i>
1	Adam Barr
2	Sean Chai
3	Eva Corets
4	Erin O'Melia

**sales**

<i>buyer_id</i>	<i>prod_id</i>	<i>qty</i>
1	2	15
1	3	5
3	1	37
4	5	11
2	2	1003

**produce**

<i>prod_id</i>	<i>prod_name</i>
1	Apples
2	Pears
3	Oranges
4	Bananas
5	Peaches

**Result**

<i>buyer_name</i>	<i>prod_name</i>	<i>qty</i>
Erin O'Melia	Apples	37
Adam Barr	Pears	15
Erin O'Melia	Pears	1003
Adam Barr	Oranges	5
Eva Corets	Peaches	11

# Self Joining

```
USE joindb
SELECT a.buyer_id AS buyer1, a.prod_id
      ,b.buyer_id AS buyer2
FROM sales AS a
JOIN sales AS b
  ON a.prod_id = b.prod_id
WHERE a.buyer_id > b.buyer_id
GO
```

**sales a**

<i>buyer_id</i>	<i>prod_id</i>	<i>qty</i>
1	2	15
1	3	5
4	1	37
3	5	11
4	2	1003

**sales b**

<i>buyer_id</i>	<i>prod_id</i>	<i>qty</i>
1	2	15
1	3	5
4	1	37
3	5	11
4	2	1003

**Result**

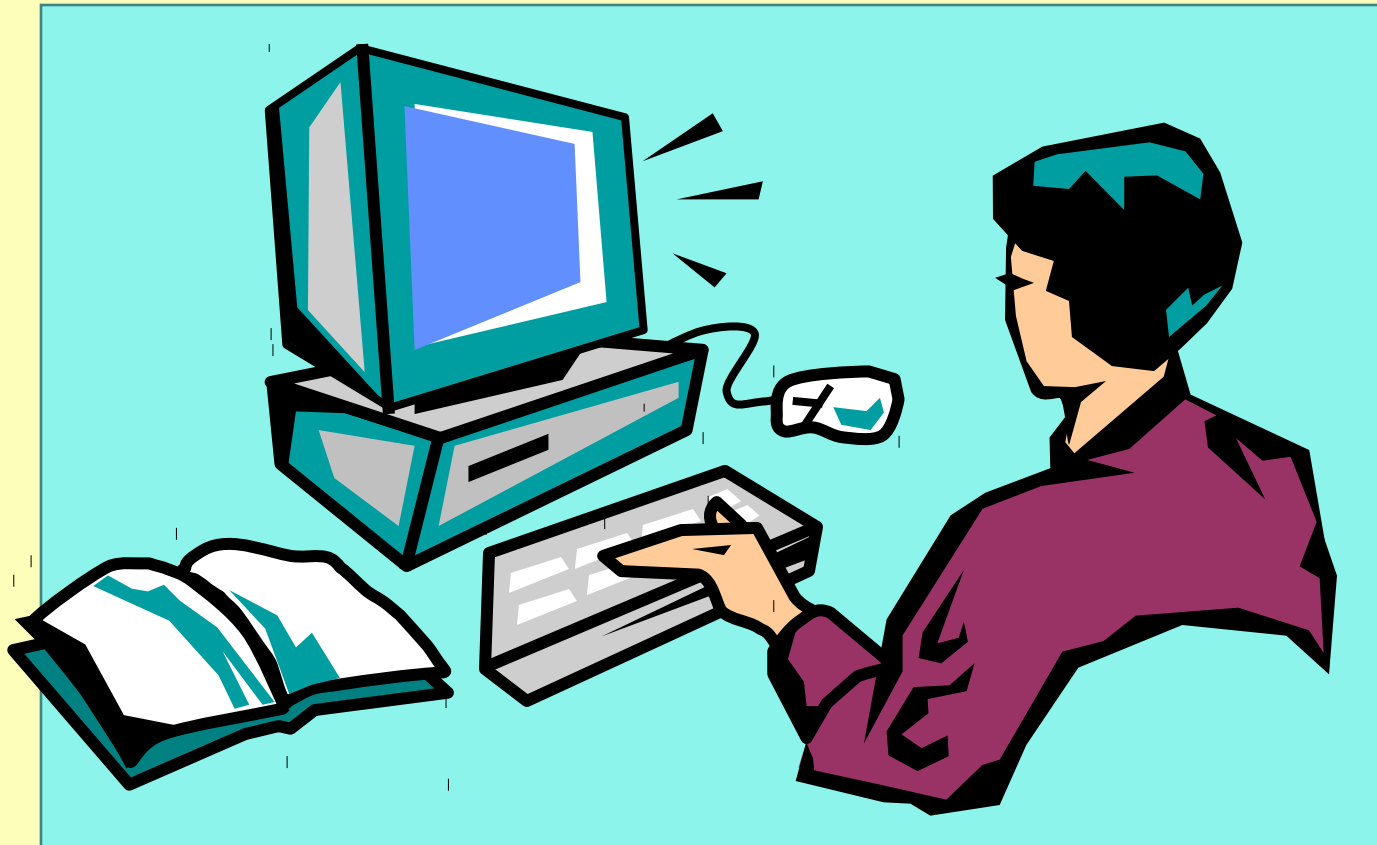
<i>buyer1</i>	<i>prod_id</i>	<i>buyer2</i>
4	2	1

# 以 UNION 上下整併資料表

- 利用 **UNION**，可將兩個結果集上下整併成一個結果集。
- 注意事項：
  - 資料型態必須相仿
  - 欄位數量必須相同
  - 系統預設會自動剔除內容相同的記錄

```
USE northwind
SELECT  (firstname + ' ' + lastname) AS name
        ,city, postalcode
FROM employees
UNION
SELECT  companyname, city, postalcode
FROM customers
GO
```

# 練習：Join 與 Union





# 活性子查詢

# 活用子查詢 ( Subquery )

- 子查詢 ( Subquery ) 簡介
- 以子查詢的結果為基礎資料表
- 將「子查詢」視為運算式
- 子查詢以資料表別名參用外部查詢的內容
- 子查詢聯合 IN 子句的寫法

# 子查詢 ( Subquery ) 簡介

- 在一個查詢指令中，內嵌另一個查詢指令。
- 內嵌的子查詢必須前後以括號框住
- 如此，可將複雜的查詢分解成數個小查詢步驟
- 以某個查詢的結果為基礎，解決進一步的問題
- 例如，條列出最後一天的訂單：

```
USE northwind
SELECT *
  FROM Orders
 WHERE OrderDate =
        (SELECT MAX(OrderDate) FROM Orders)
GO
```

# 以子查詢的結果為基礎資料表

- 將「子查詢的結果」視同是一個資料表。
- 必須加上資料表別名。

```
USE northwind
SELECT T.orderid, T.customerid
FROM ( SELECT orderid, customerid
        FROM orders ) AS T
GO
```

# 將「子查詢」視為運算式

- 凡是允許運算式的地方，都可以利用括號內嵌「子查詢」。

```
USE pubs
SELECT title, price
      , ( SELECT AVG(price) FROM titles) AS average
      , price-(SELECT AVG(price) FROM titles) AS difference
FROM titles
WHERE type='popular_comp'
GO
```

# 子查詢以資料表別名參用外部查詢的內容

**1** 外部查詢將欄位值傳入內部查詢

```
USE northwind
SELECT orderid, customerid
FROM orders AS or1
WHERE 20 < (SELECT quantity
            FROM [order details] AS od
            WHERE od.orderid = or1.orderid
            AND   od.productid = 23)

GO
```

**2** 內部查詢以傳入值進行資料查詢

**3** 內部查詢傳回結果到外部查詢

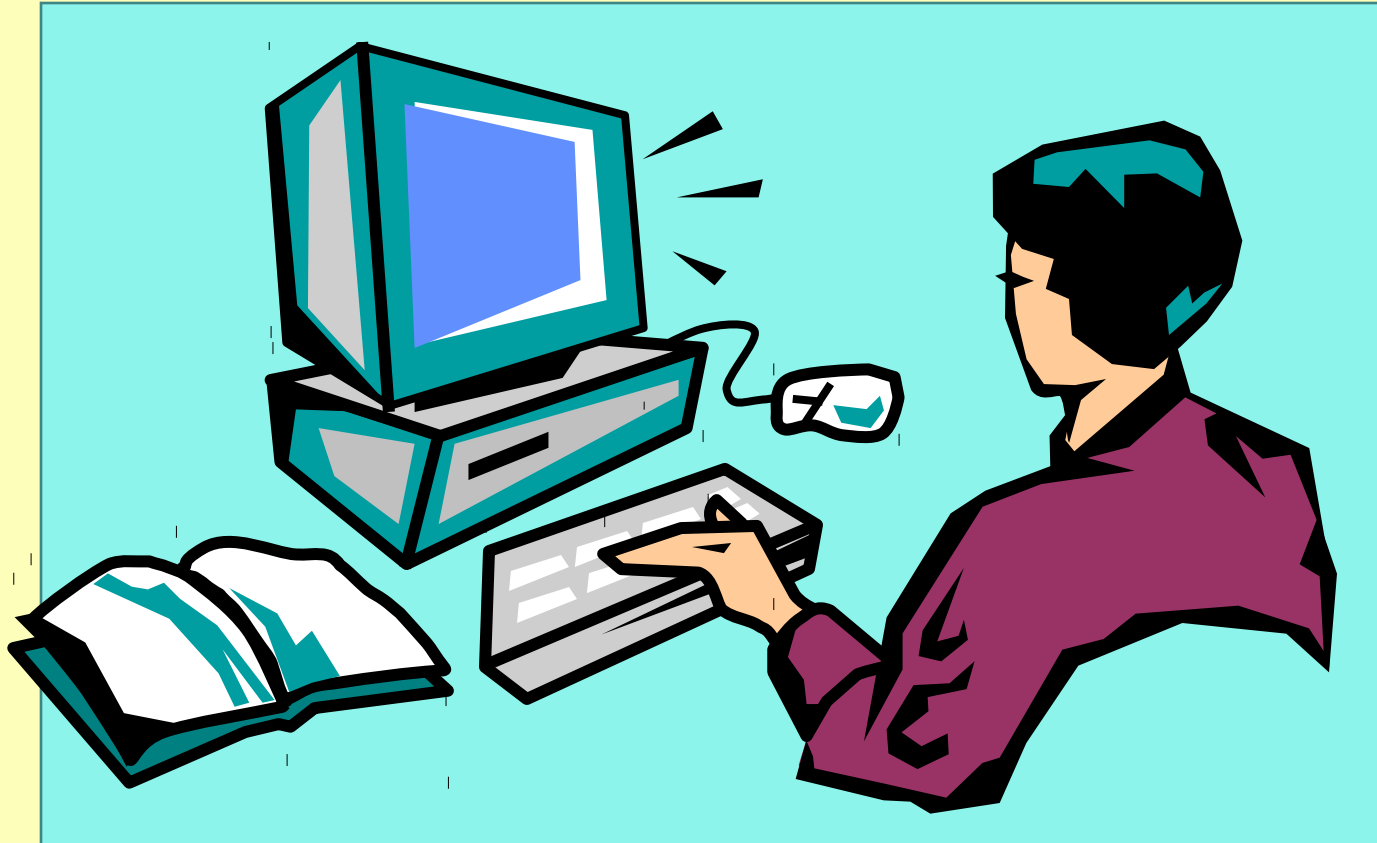
**4** 針對外部查詢的每一筆都重複 1~3 的步驟

# 子查詢聯合 IN 子句的寫法

- 有時候，想查 A 資料表時，得靠 B 資料表幫忙。
- 先從訂單明細查出哪些訂單編號「有訂購 23 號商品而且訂超過 20 個」，再以此為基礎，列出這些編號的訂單資料。

```
select * from orders
  where OrderID in
    (
      select OrderID
        from [Order Details]
       where ProductID = 23 and Quantity > 20
    )
```

# 練習：子查詢





**修改資料**

**Insert 、 Update 、 Delete**

# 修改資料

## ■ Insert 語法說明

- Insert 新增單筆資料
- Insert ... Select
- Select ... Into

## ■ Delete 語法說明

## ■ Update 語法說明

# 新增單筆資料

- 將各個欄位以逗號分隔條列
- 再利用 VALUES 一一對應各欄位值
- 資料必須符合資料型態並且通過檢查規則（例如：主鍵鍵值不可重複），資料才可進入資料表。
- 自動編號欄位以及計算型欄位不能指定欄位值。

```
USE northwind
INSERT customers
    (customerid, companyname, contactname, contacttitle
    ,address, city, region, postalcode, country, phone
    ,fax)

VALUES ('PECOF', 'Pecos Coffee Company', 'Michael Dunn'
    , 'Owner', '1900 Oak Street', 'Vancouver', 'BC'
    , 'V3F 2K1', 'Canada', '(604) 555-3392'
    , '(604) 555-7293')

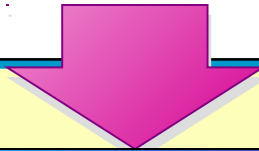
GO
```

# 若略過特定欄位的值，將代入預設值或 Null

新增資料時，沒有指定「電話」的欄位內容...

```
USE northwind
INSERT shippers (companymame)
VALUES ('Fitch & Mather')
GO
```

```
USE northwind
SELECT *
FROM shippers
WHERE companymame = 'Fitch & Mather'
GO
```



<i>shipperid</i>	<i>companymame</i>	<i>phone</i>
37	Fitch & Mather	Null

# INSERT...SELECT 敘述

- 先利用 **SELECT** 敘述，從「來源資料表」提出資料，然後將之匯入到「目的資料表」。
- 「目的資料表」必須是現有存在的資料表。
- 必須配合「目的資料表」，依序提供資料型態相容的欄位值。

```
USE northwind
INSERT customers
  SELECT substring(firstname, 1, 3)
         + substring (lastname, 1, 2)
         ,lastname, firstname, title, address, city
         ,region, postcode, country, homephone, NULL
FROM employees
GO
```

# SELECT ... INTO 敘述

- 以 **SELECT** 敘述的結果集，建立一個全新的資料表。
- 目的資料表若已經存在，**SELECT...INTO** 敘述將無法執行。
- 目的資料表名稱之前若加上 **#** 符號，表示新資料表是一個臨時性資料表，結束連線時，該資料表會自動消失。

```
USE northwind
SELECT productname AS products
      ,unitprice AS price
      ,(unitprice * 1.1) AS tax
INTO #pricetable
FROM products
GO
```

## ◆ 刪除資料

- **DELETE** 語法說明
- **TRUNCATE TABLE**

# DELETE 敘述

- 利用 **DELETE** 敘述可刪除符合 **WHERE** 條件的資料。
- 若是沒寫 **WHERE**，等同於刪除整份資料表的內容

```
USE northwind
DELETE orders
  WHERE DATEDIFF(MONTH, shippeddate, GETDATE()) >= 6
GO
```



# TRUNCATE TABLE 敘述

- **TRUNCATE TABLE** 敘述會刪除整份資料表的內容。
- 資料表的內容全部清空，資料表的資料結構仍在。
- 由於 **TRUNCATE TABLE** 敘述刪除資料時並不會進行 **LOG** 記錄，因此將無法以 **ROLLBACK** 回復資料。

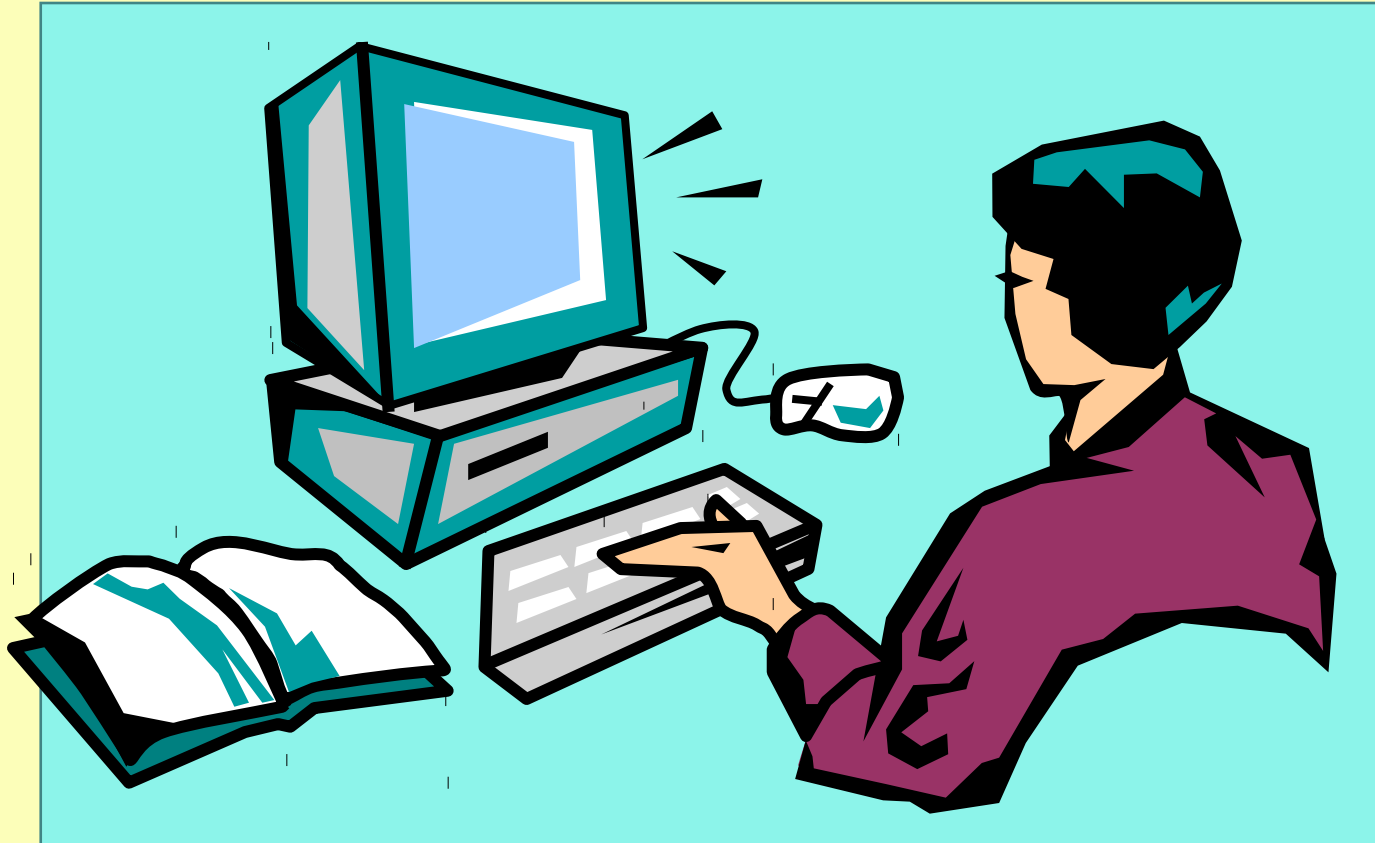
```
USE northwind
TRUNCATE TABLE orders
GO
```

# 以 Update 敘述修改資料

- 以 **WHERE** 子句指定要修改的是哪些記錄
- 以 **SET** 設定新欄位值
- 新欄位值必須通過各項 **Constraint** 的資料檢驗
- 建議加上開放式並行檢查，將舊值也融入 **WHERE**，以確保沒有與其他使用者發生資料衝突。

```
USE northwind
UPDATE products
  SET unitprice = (unitprice * 1.1)
  WHERE CategoryID = 1
GO
```

# 練習：新增、修改、刪除資料

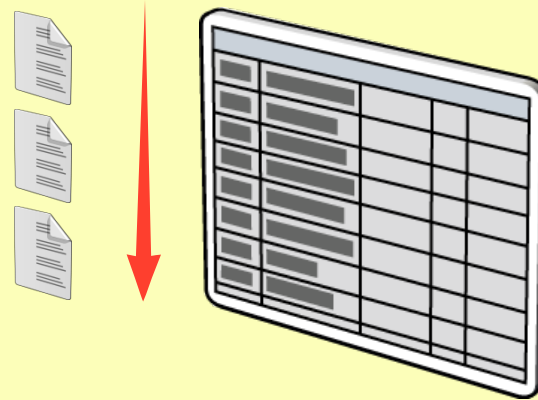


**善用索引加快查詢效能**

# SQL Server 如何讀取資料

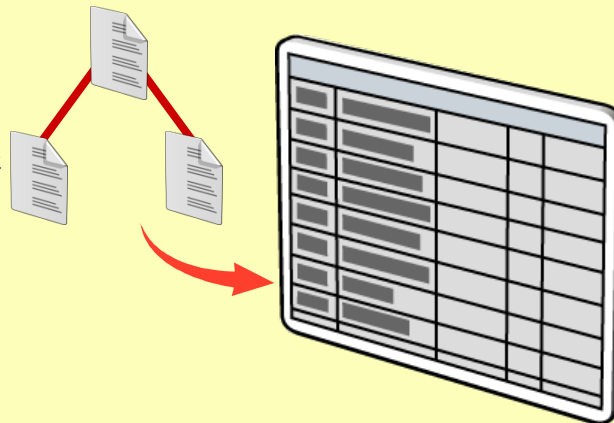
- 沒有索引時...

- SQL Server 將全部的資料都讀一遍，稱為 **Table Scan**



- 索引 (Index):

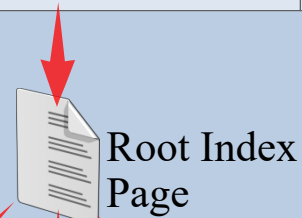
- SQL Server 利用索引提供的線索，有系統的縮小範圍進行 **Seek**。



# 什麼是叢集索引 ( Clustered Index ) ?

sys.partitions

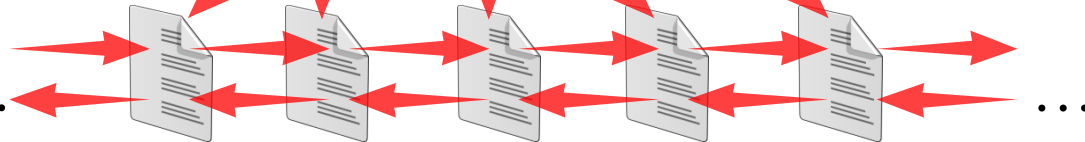
id	index_id = 1	root_page	
----	--------------	-----------	--



Intermediate Level  
Index Pages



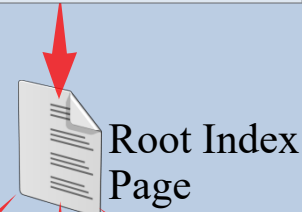
Leaf Nodes  
Data Pages ...



# 什麼是非叢集索引 (Nonclustered Index)?

sys.partitions

id	index_id > 1	root_page	
----	--------------	-----------	--



Leaf Nodes  
Index Pages



Heap or  
Clustered  
Index  
Data Pages



# 索引語法說明

```
CREATE [ UNIQUE ] [ CLUSTERED | NONCLUSTERED ]  
INDEX index_name ON { table | view } ( column [ ASC |  
DESC ] [ ,...n ] )  
INCLUDE ( column [ ,...n ] )  
[WITH option [ ,...n ] ]  
[ON {partition_scheme (column) | filegroup | "default" } ]
```


WITH 選項	用途
ALLOW_ROW_LOCKS	可 / 否進行單筆鎖定
ALLOW_PAGE_LOCKS	可 / 否進行單頁鎖定
ONLINE	索引編製期間，資料表是 / 否在線 (online)
FILLFACTOR	leaf-level 資料頁的內容佔該頁的百分比
PAD_INDEX	non-leaf-level 資料頁也比照上述 FillFactor 的值進行空間管理。



# 加上 Unique ，確保欄位值不會重複

## 可確保欄位值不會重複

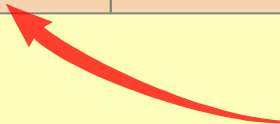
```
CREATE UNIQUE NONCLUSTERED INDEX [AK_Employee_LoginID]  
ON [HumanResources].[Employee] ( [LoginID] ASC)
```



EmployeeID	LoginID	Gender	MaritalStatus	...
216	mike0	M	S	...
231	fukiko0	M	M	...
242	pat0	M	S	...

...

251	pat0	F	S	...
-----	------	---	---	-----

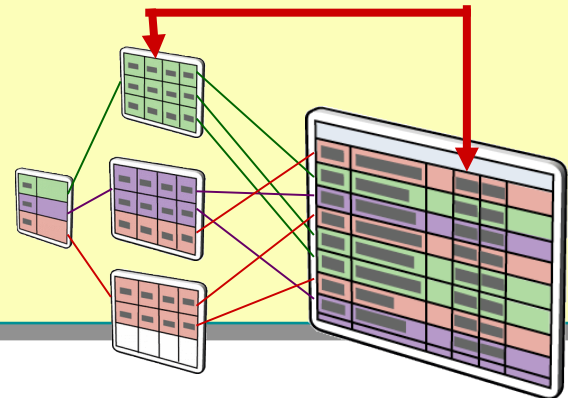


索引鍵值不許重複

# 索引的 INCLUDE 子句

- 以 Include 子句，將「**Select** 欄位清單所需的欄位」含入到非叢集索引的資料頁。
- 於是，**Select** 敘述執行時，非叢集索引早已有 **Select** 敘述所需的資料，**SQL Server** 因此不需再到叢集索引找資料，藉此提高查詢效能。

```
CREATE NONCLUSTERED INDEX AK_Employee_LoginID  
ON HumanResources.Employee ( LoginID ASC)  
INCLUDE ( ContactID, NationalIDNumber)
```

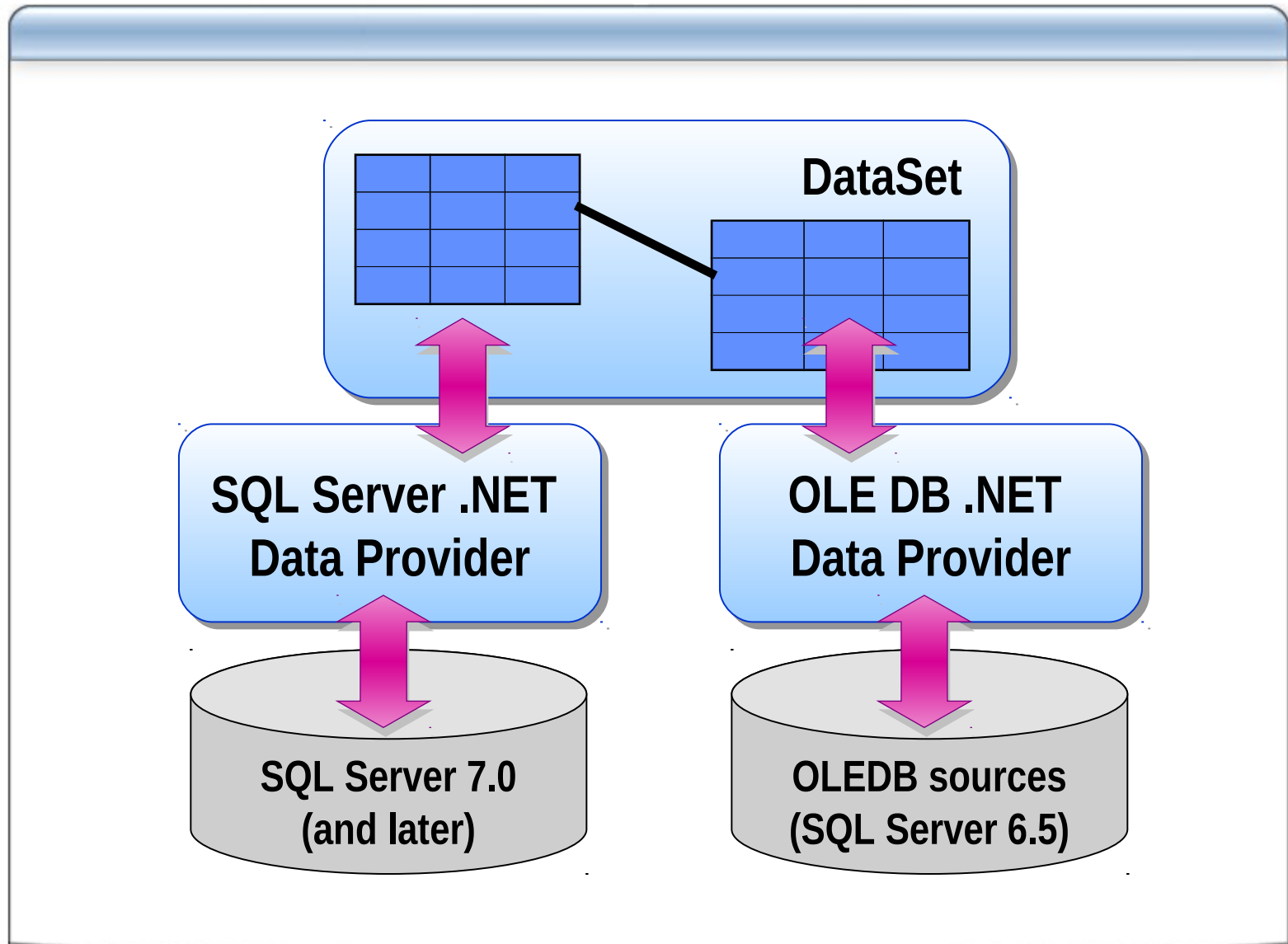


# ADO.NET 資料存取

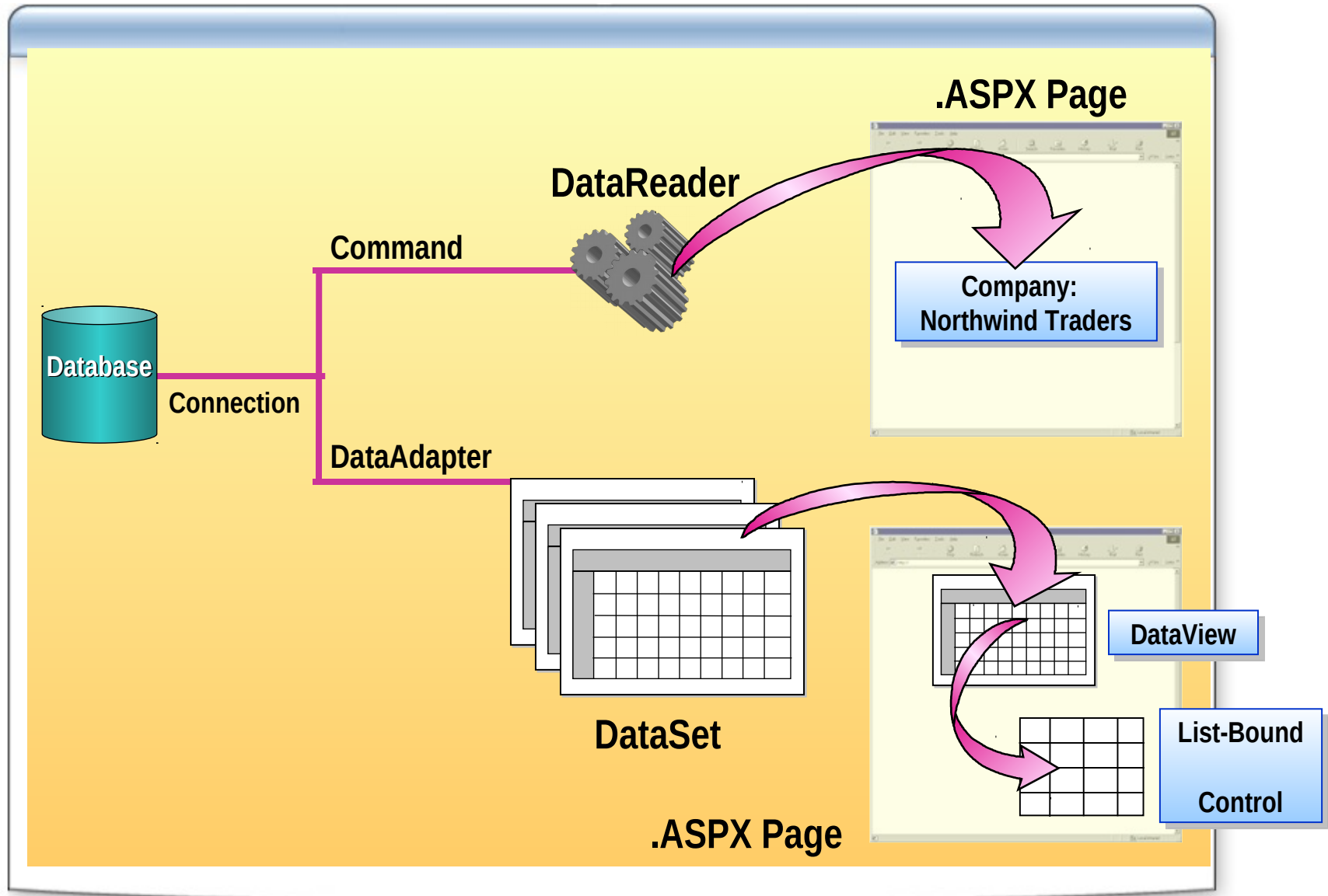
# 內容大綱

- ADO.NET 物件模型
- 使用 ADO.NET 存取資料
- 建立連線物件
- 使用資料配接器 (DataAdapter)
- 使用資料集 (DataSet)
- DataView
- 資料繫結
- 以 DataReader 讀取資料
- 使用 ADO.NET Command 物件送出 SQL 指令
- 使用 Command 物件的參數集合
- 呼叫預儲程序 (Stored Procedures)
- 異動處理 (Transaction)

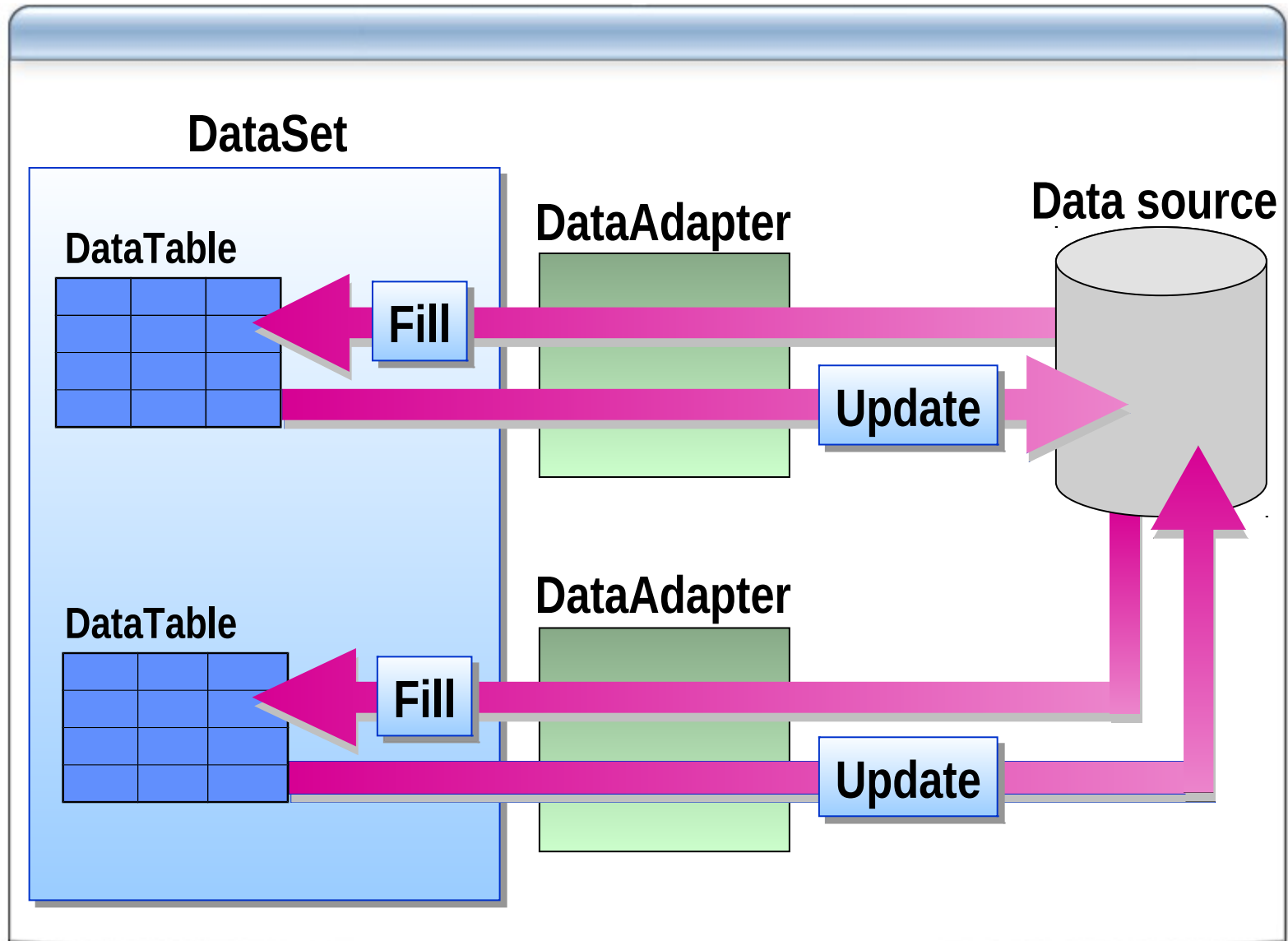
# ADO .NET 物件模型 (一)



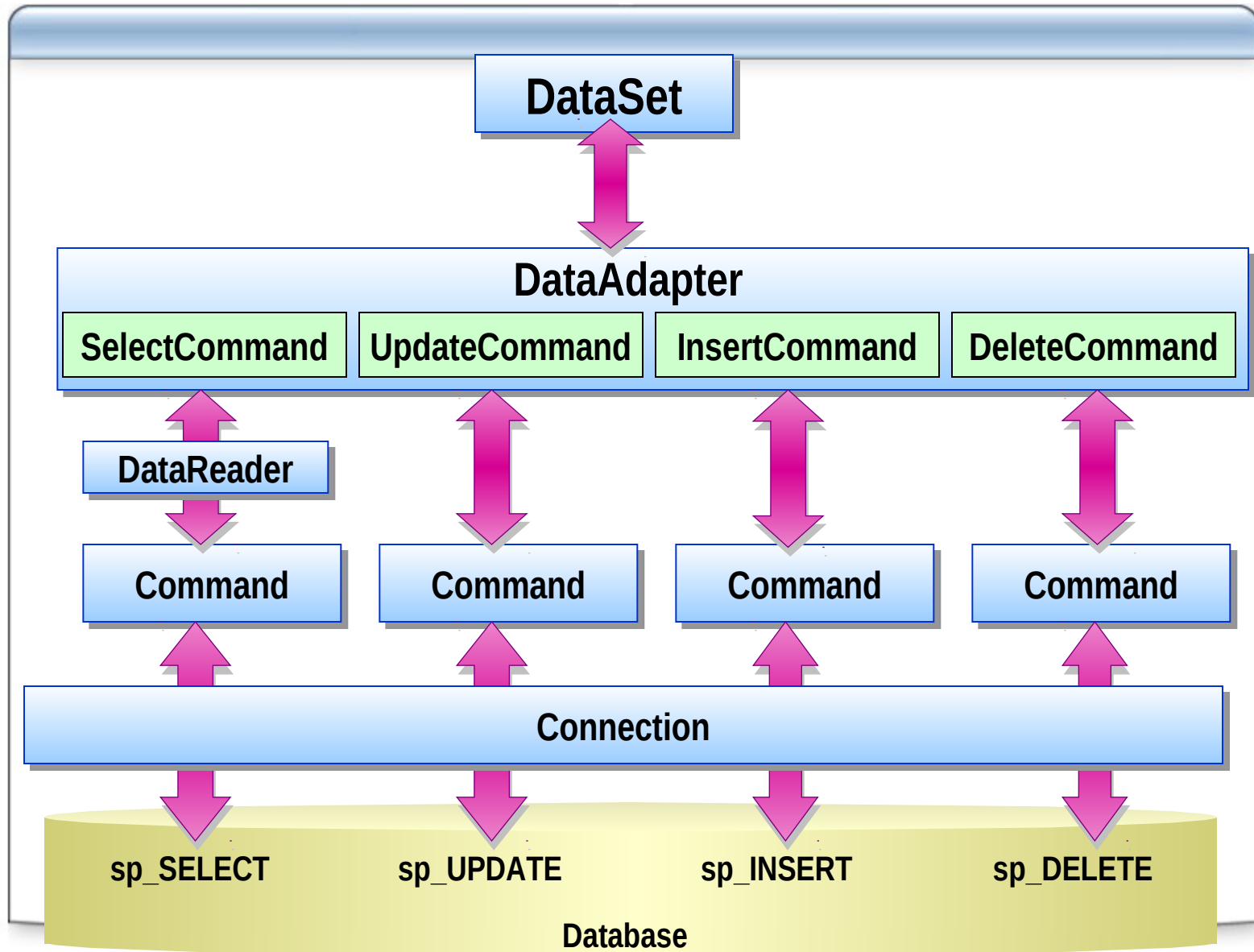
# ADO .NET 物件模型 (二)



# 什麼是資料配接器 DataAdapter?

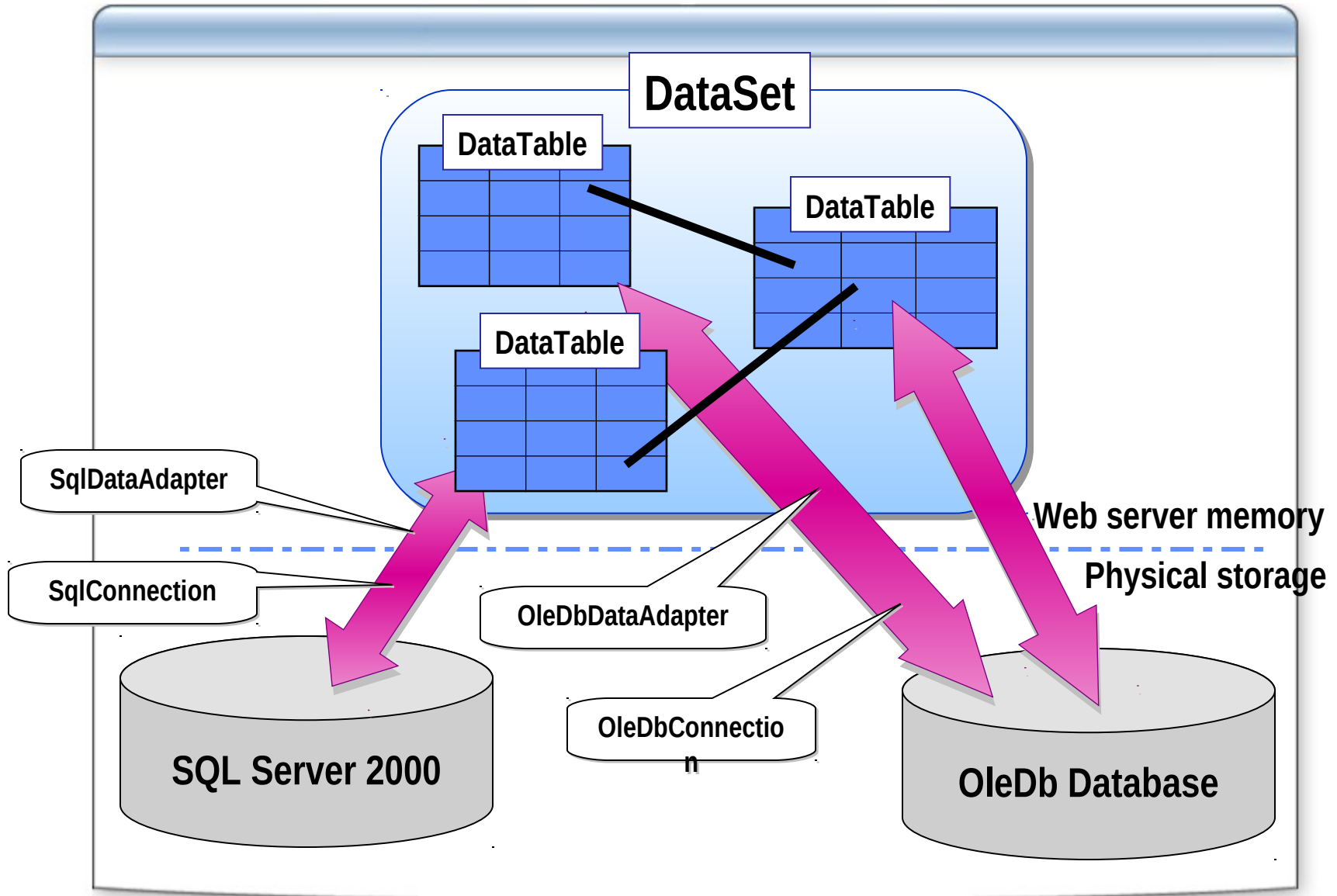


# 資料配接器的物件模型



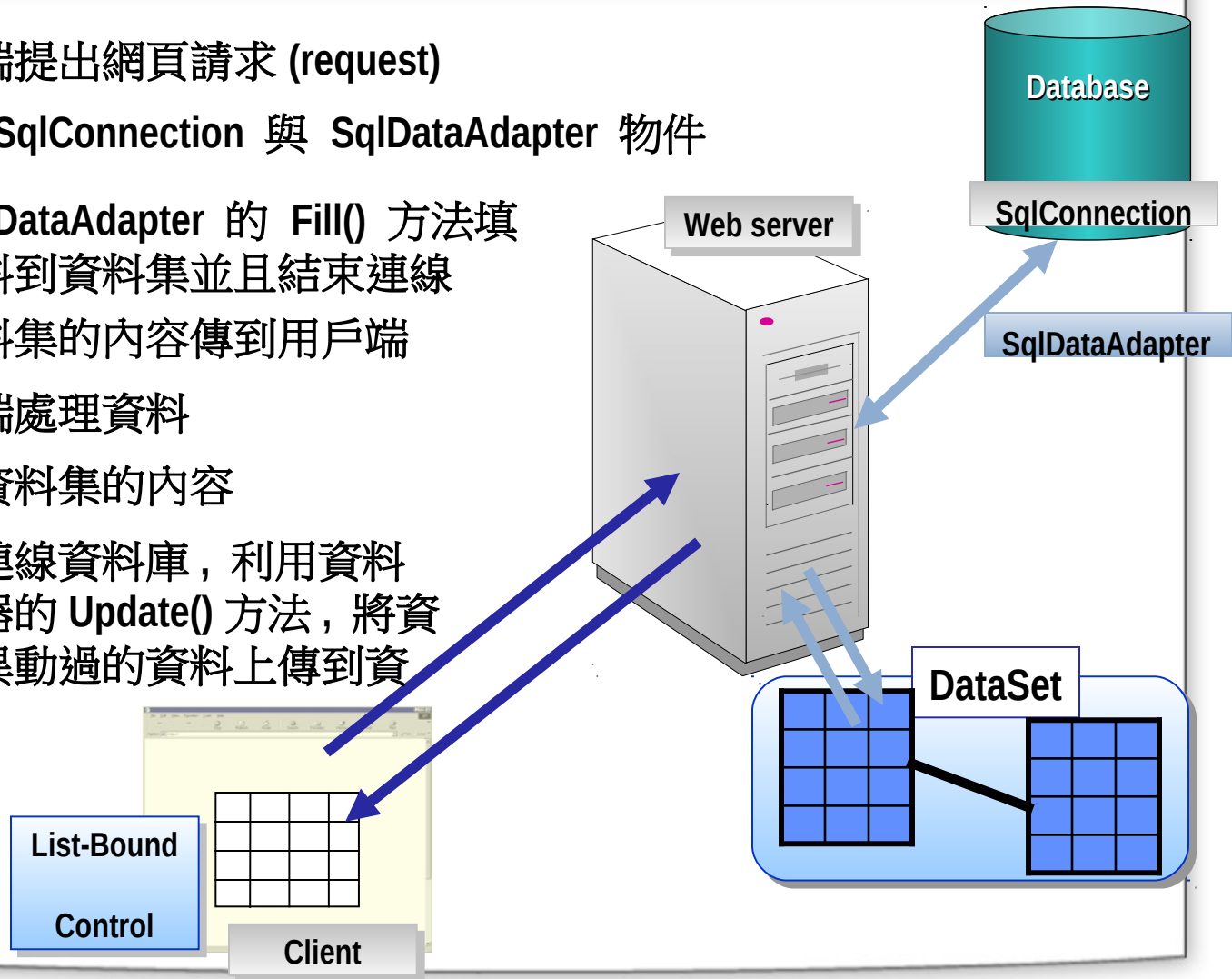


# 什麼是資料集？

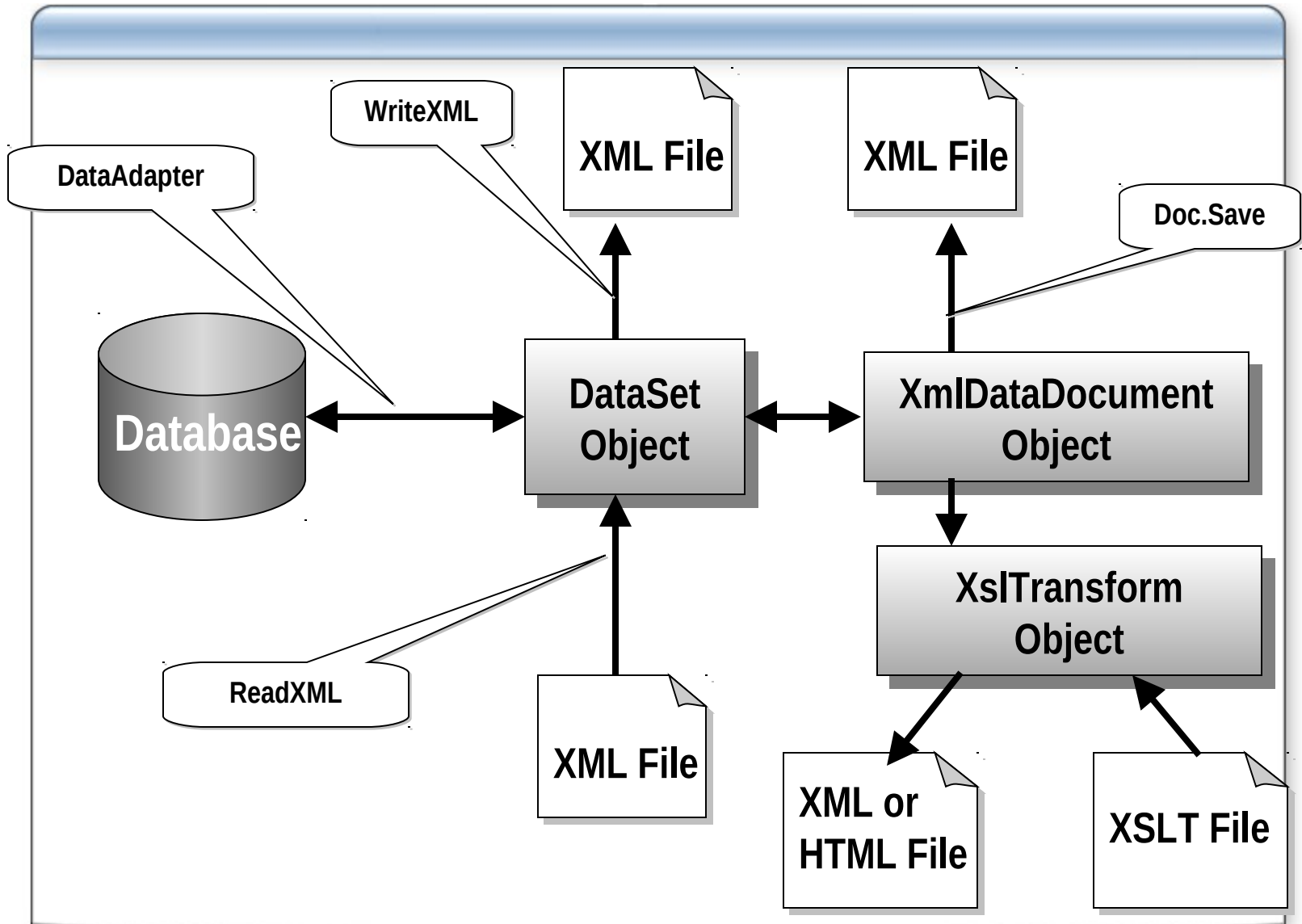


# 使用 ADO.NET 存取資料

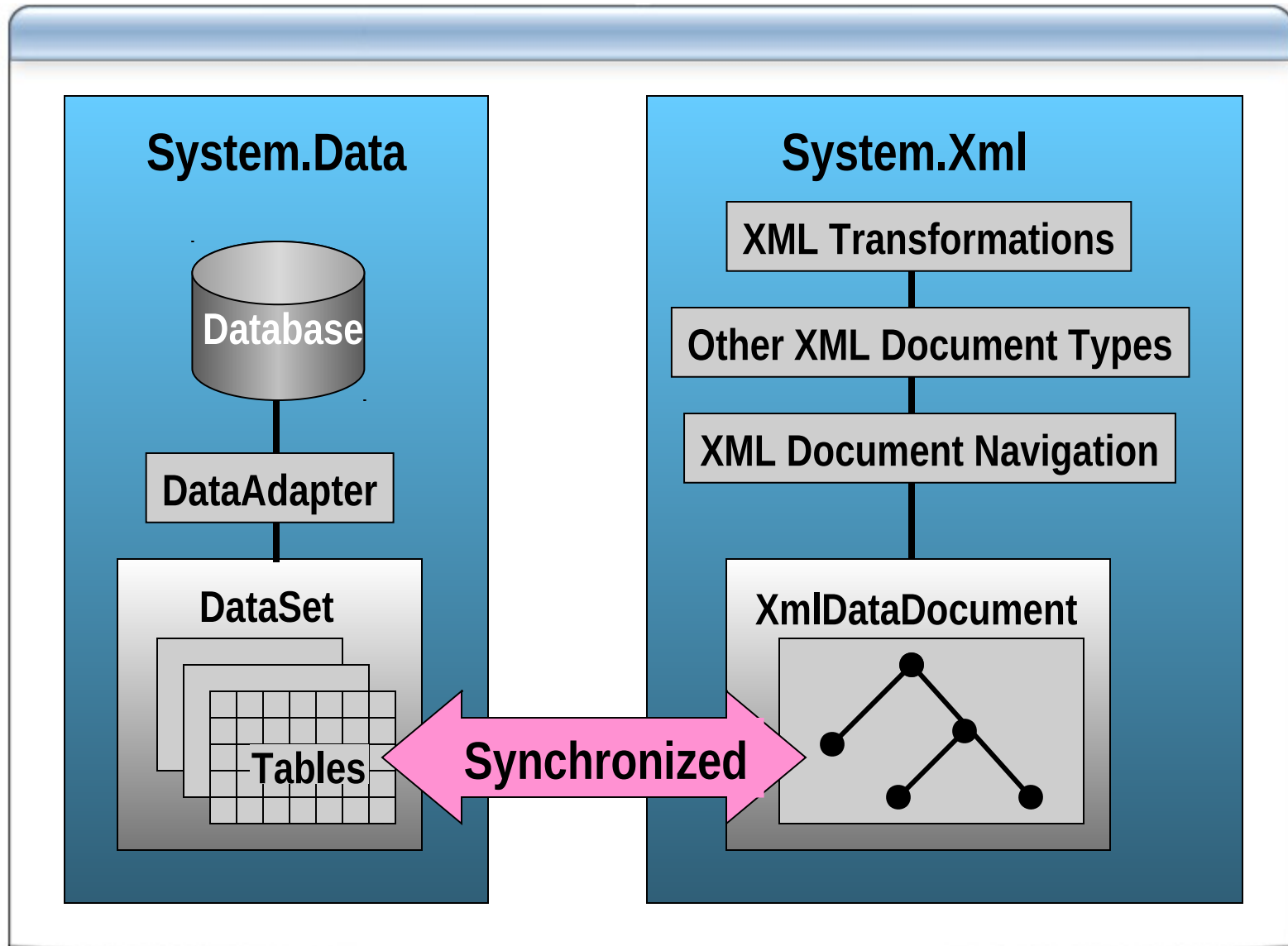
- 1 用戶端提出網頁請求 (request)
- 2 建立 SqlConnection 與 SqlDataAdapter 物件
- 3 呼叫 DataAdapter 的 Fill() 方法填入資料到資料集並且結束連線
- 4 將資料集的內容傳到用戶端
- 5 用戶端處理資料
- 6 異動資料集的內容
- 7 重新連線資料庫，利用資料配接器的 Update() 方法，將資料集異動過的資料上傳到資料庫



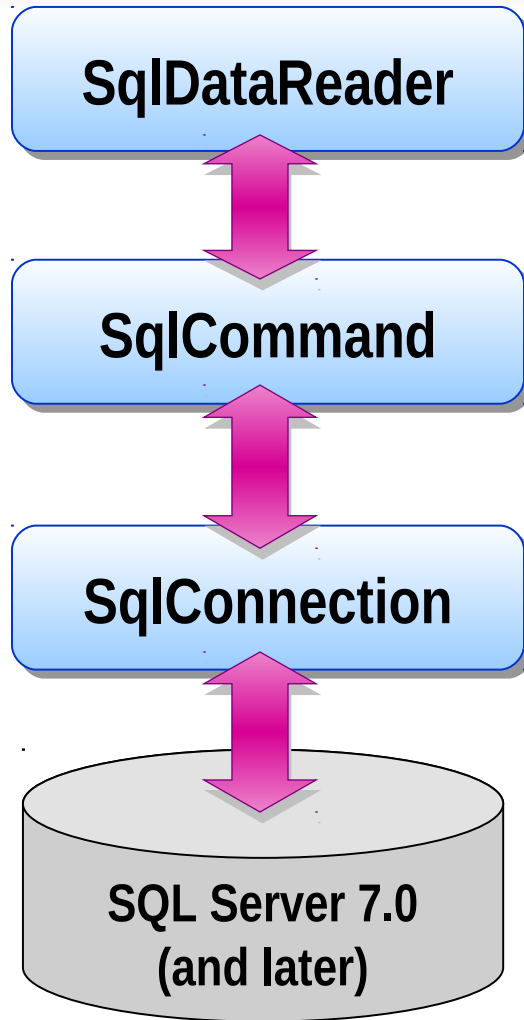
# XML 與資料集 (Dataset) 的關係 (一)



# XML 與資料集 (Dataset) 的關係 (二)



# 以 SqlDataReader 在連線情況下讀取資料



1. `Connection.Open()` 開啟連線
2. `SqlCommand.ExecuteReader()` 送出 SQL 指令並取得 DataReader 物件
3. 連續呼叫 DataReader 的 `Read()` 方法逐筆讀出資料
4. `DataReader.Close()` 關閉 DataReader
5. `Connection.Close()` 結束連線

# 比較 DataSet 與 DataReader

DataSet	DataReader
讀 / 寫模式	唯讀模式
可內含不同資料庫的資料表	以 SQL 敘述從單一資料庫調出資料
離線時，仍可存取資料集	必須連線才能讀資料
可資料繫結到多個控制項	資料僅繫結到單一控制項
處理模式為：往返一趟	單向進行
處理速度較慢	處理速度較快
Visual Studio .NET 提供許多自動產生程式的工具	手工撰寫程式

# 建立連線物件

- 建立連線物件時一併指定 **ConnectionString** 的內容

```
string strConn = "data source=localhost; " +  
    "initial catalog=northwind; integrated security=true";  
SqlConnection conn = new SqlConnection(strConn);
```

- 連線字串中的參數：

- Connection timeout
- Data source
- Initial catalog
- Integrated security
- Password
- Persist security info
- Provider
- User ID

# 什麼是連線集區 (Connection Pooling)?

**連線集區 = 重複使用連線物件**

- 連線安全設定相同時，.NET 內建的連線集區功能會重複使用連線物件。

**連線集區如何運作：**

- 應用程式嘗試開啟資料庫連線
- 如果連線集區有閒置可用連線，.NET 將傳回該可用連線給應用程式
- 如果連線集區沒有可用連線，則建立新的連線。
- 關閉連線時，該連線物件自動返回連線集區成為可重複使用的閒置連線



# 使用資料配接器 (DataAdapter)

- 建立資料配接器時一併傳入資料查詢 SQL 敘述

```
SqlDataAdapter da = new SqlDataAdapter  
    ("select * from Authors", conn);
```

- 後來可修改 SelectCommand 屬性：

```
da.SelectCommand.CommandText;  
da.SelectCommand.Connection;
```

- 視程式需要，繼續設定 InsertCommand, UpdateCommand 與 DeleteCommand 等屬性

# 使用資料集 (DataSet)

- 建立並且導入資料到資料表 (DataTable)

- 呼叫資料配接器的 Fill() 方法以執行 SelectCommand 的 SQL 敘述

```
DataSet ds = new DataSet();  
da.Fill(ds, "Authors");
```

- 存取資料表物件 (DataTable) 的內容

```
ds.Tables["Authors"].Rows.Count;
```

```
string str="";  
  
foreach(DataRow r in  
    ds.Tables["Authors"].Rows)  
{  
    str += r[2];  
    str += r["au_lname"];  
}
```

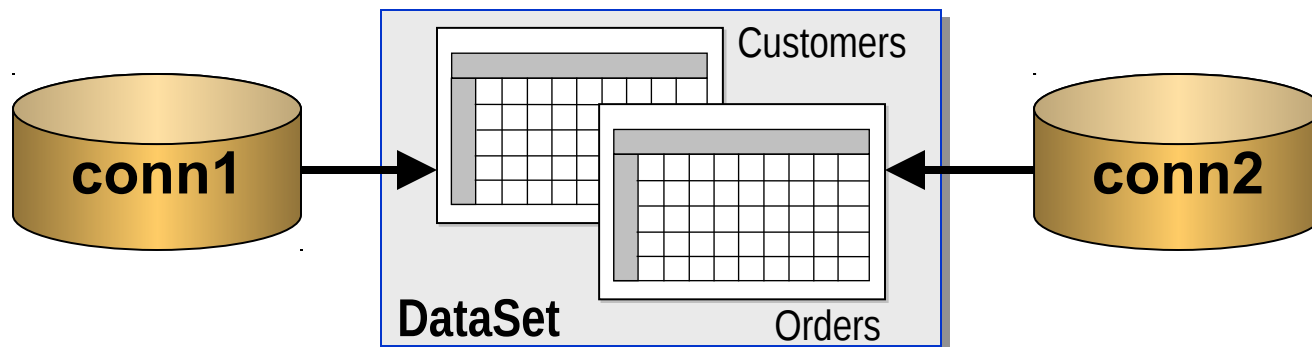
# 資料集可內含多個資料表

- 加入第一個資料表

```
daCustomers = New SqlDataAdapter _  
    ("select * from Customers", conn1)  
daCustomers.Fill(ds, "Customers")
```

- 加入後續的資料表

```
daOrders = New SqlDataAdapter _  
    ("select * from Orders", conn2)  
daOrders.Fill(ds, "Orders")
```



# 建立資料表之間的關聯關係

- 找出父階欄位

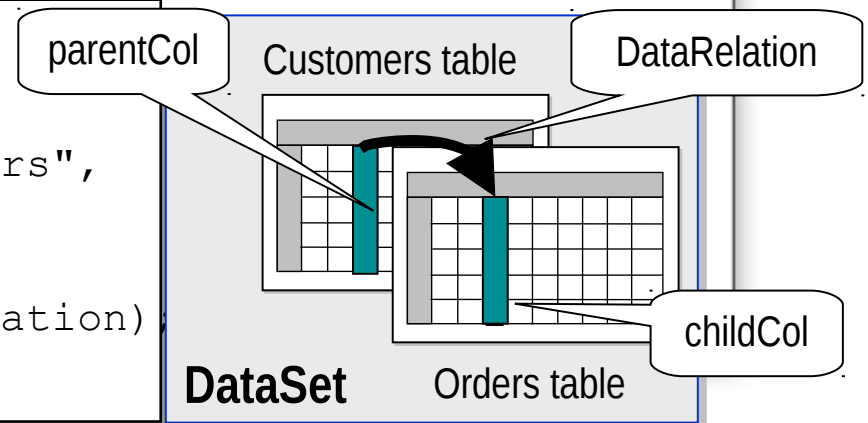
```
parentColumn =  
    myDataSet.Tables["Publishers"].Columns["pub_id"];
```

- 找出子階欄位

```
childColumn =  
    myDataSet.Tables["Titles"].Columns["pub_id"];
```

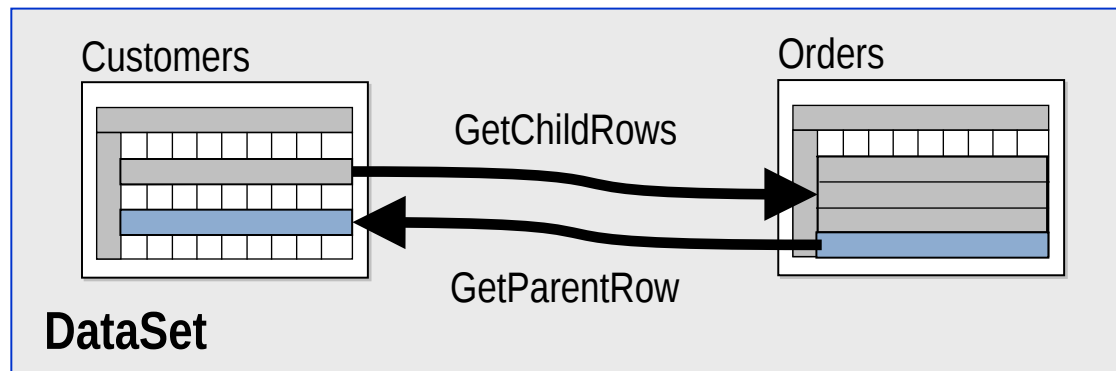
- 以父，子兩欄位建立 DataRelation 物件

```
myDataRelation =  
    new DataRelation("TitlePublishers",  
        parentColumn, childColumn);  
myDataSet.Relations.Add(myDataRelation);
```



# 找出對應的子階記錄

```
ds.Tables[index].Rows[index].GetChildRows("relation");  
ds.Tables[index].Rows[index].GetParentRow("relation");
```

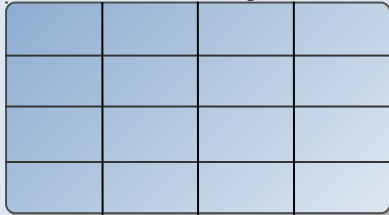


# 什麼是 DataView?

**DataView 是資料表的特定觀點。**

**DataSet object**

**DataTable object**




**DefaultView**

**DataView object**

**Default sort / filter  
criteria**

**DataView objects**

**Alternative sort / filter  
criteria**

**Additional views**



# 使用 DataView

- 資料表的 **DefaultView** 屬性所傳回的 **DataView**, 預設為原排序次序的全部記錄

```
DataView dv = ds.Tables["Authors"].DefaultView;
```

- 建立新的 **DataView** 並且指定不同的過濾條件

```
DataView dv = new DataView(ds.Tables["Authors"]);  
dv.RowFilter = "state = 'CA'";
```

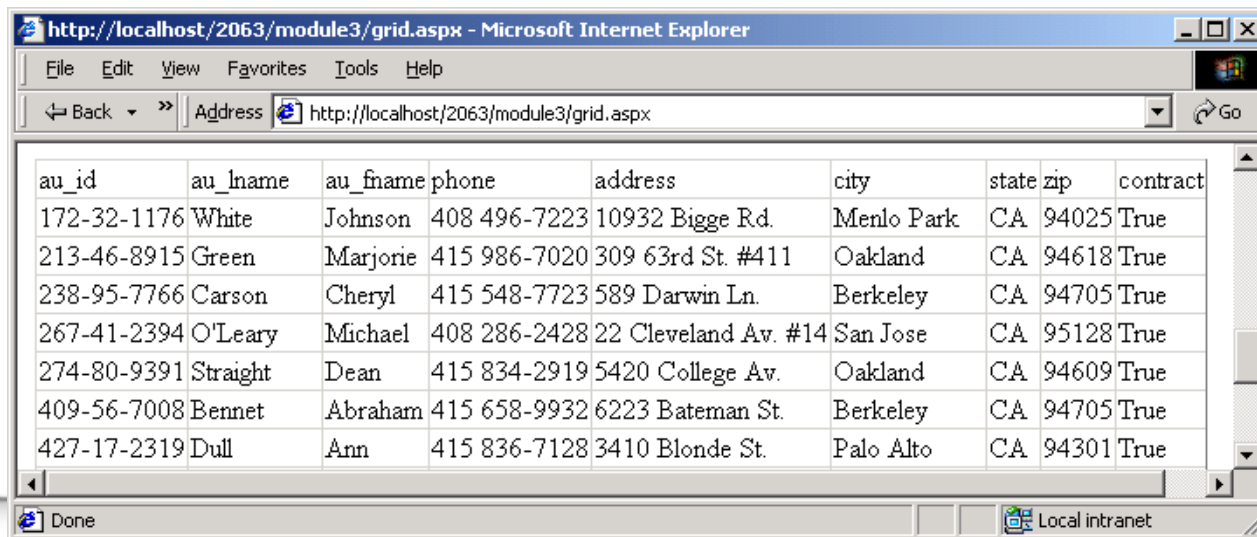
# 資料繫結

- 放置 / 建立控制項

```
<asp:DataGrid id="dg" runat="server" />
```

- 繫結到資料表或 DataView

```
dg.DataSource = ds;  
dg.DataMember = "Authors";  
dg.DataBind();
```

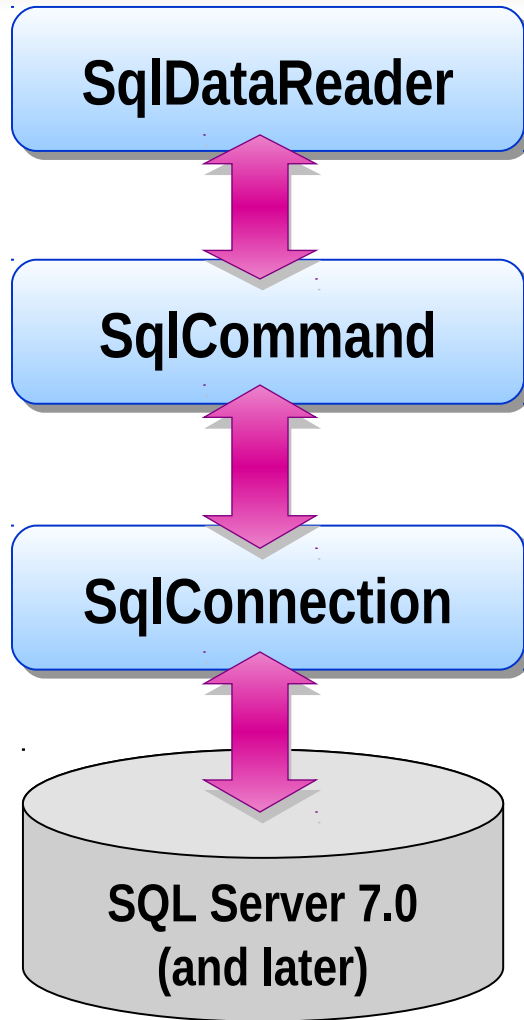


The screenshot shows a Microsoft Internet Explorer window with the address bar displaying `http://localhost/2063/module3/grid.aspx`. The page content is a data grid with 9 columns: `au_id`, `au_lname`, `au_fname`, `phone`, `address`, `city`, `state`, `zip`, and `contract`. The grid contains 7 rows of author data.

au_id	au_lname	au_fname	phone	address	city	state	zip	contract
172-32-1176	White	Johnson	408 496-7223	10932 Bigge Rd.	Menlo Park	CA	94025	True
213-46-8915	Green	Marjorie	415 986-7020	309 63rd St. #411	Oakland	CA	94618	True
238-95-7766	Carson	Cheryl	415 548-7723	589 Darwin Ln.	Berkeley	CA	94705	True
267-41-2394	O'Leary	Michael	408 286-2428	22 Cleveland Av. #14	San Jose	CA	95128	True
274-80-9391	Straight	Dean	415 834-2919	5420 College Av.	Oakland	CA	94609	True
409-56-7008	Bennet	Abraham	415 658-9932	6223 Bateman St.	Berkeley	CA	94705	True
427-17-2319	Dull	Ann	415 836-7128	3410 Blonde St.	Palo Alto	CA	94301	True



# 以 SqlDataReader 讀取資料



1. `Connection.Open()` 開啟連線
2. `SqlCommand.ExecuteReader()`  
送出 SQL 指令並取得  
DataReader 物件
3. 連續呼叫 DataReader 的  
`Read()` 方法逐筆讀出資料
4. `DataReader.Close()`  
關閉 DataReader
5. `Connection.Close()`  
結束連線

# 以 DataReader 讀取資料

- 呼叫 Read() 方法以取得一筆記錄
  - 若傳回 false 值，表示已讀完資料
- 讀取各欄位值
  - 傳入欄位編號或欄位名稱以指定欄位
  - 呼叫 GetXXX 函式的效能比較好
- 關閉 DataReader
- 結束連線

```
myReader = cmd.ExecuteReader();  
while (myReader.Read())  
{  
    str += myReader[1];  
    str += myReader["field"];  
    str += myReader.GetDateTime(2);  
}  
myReader.Close();  
objConnection.Close();
```

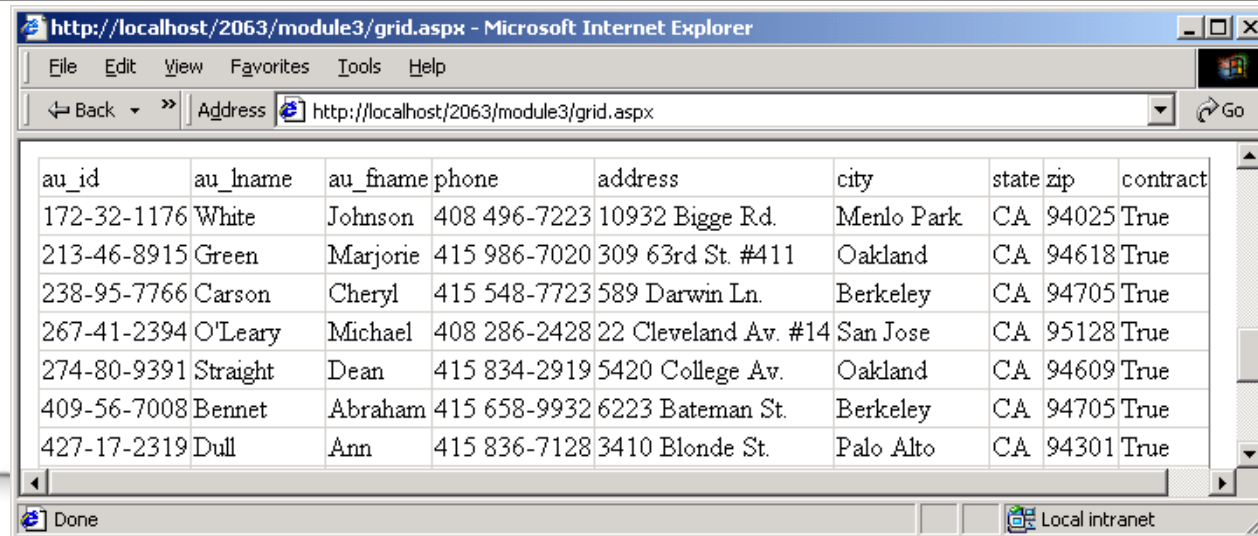
# DataReader 與控制項的資料繫結

- 置放 / 建立控制項

```
<asp:DataGrid id="dgAuthors" runat="server" />
```

- 繫結到 DataReader

```
dgAuthors.DataSource = dr;  
dgAuthors.DataBind();
```

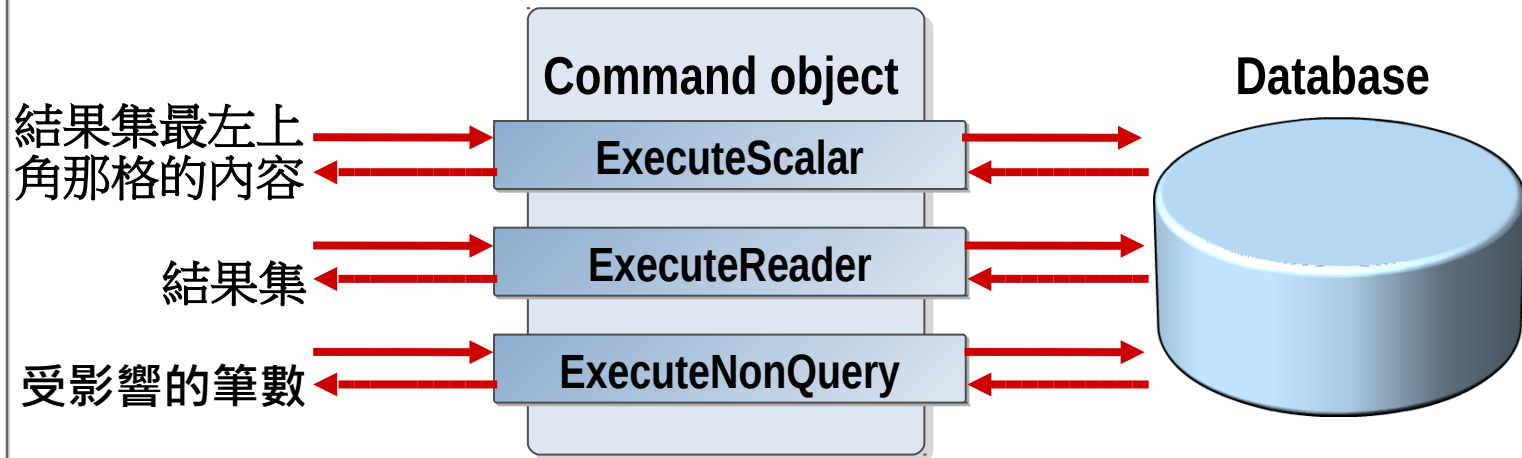


The screenshot shows a web browser window titled "http://localhost/2063/module3/grid.aspx - Microsoft Internet Explorer". The address bar shows "http://localhost/2063/module3/grid.aspx". The page content is a table with 9 columns: au\_id, au\_lname, au\_fname, phone, address, city, state, zip, and contract. The table contains 8 rows of author data.

au_id	au_lname	au_fname	phone	address	city	state	zip	contract
172-32-1176	White	Johnson	408 496-7223	10932 Bigge Rd.	Menlo Park	CA	94025	True
213-46-8915	Green	Marjorie	415 986-7020	309 63rd St. #411	Oakland	CA	94618	True
238-95-7766	Carson	Cheryl	415 548-7723	589 Darwin Ln.	Berkeley	CA	94705	True
267-41-2394	O'Leary	Michael	408 286-2428	22 Cleveland Av. #14	San Jose	CA	95128	True
274-80-9391	Straight	Dean	415 834-2919	5420 College Av.	Oakland	CA	94609	True
409-56-7008	Bennet	Abraham	415 658-9932	6223 Bateman St.	Berkeley	CA	94705	True
427-17-2319	Dull	Ann	415 836-7128	3410 Blonde St.	Palo Alto	CA	94301	True

# 使用 ADO.NET Command 物件送出 SQL 指令

使用 Command 物件可送出 SQL 敘述或呼叫預儲程序 (stored procedure)



# 使用 Command 物件的參數集合

- 1 建立參數物件
- 2 設定參數物件的屬性值
- 3 將參數物件加入 Command 的 Parameters 集合
- 4 執行 ExecureRead() 或 ExecuteNonQuery()
- 5 讀取 Output 型參數的參數值

# 使用參數 (一)

- 建立參數物件
- 設定參數物件的方向與參數值
- 將參數物件加入參數集合

```
SqlParameter param = new SqlParameter  
    ("@Beginning_Date", SqlDbType.DateTime);  
param.Direction = ParameterDirection.Input;  
param.Value = Convert.ToDateTime  
    (txtStartDate.Text);  
da.SelectCommand.Parameters.Add(param);
```

# 使用參數 (二)

- 建立參數物件，設定參數物件的方向與參數值，將參數物件加入參數集合

```
param = new SqlParameter("@ItemCount", SqlDbType.Int);  
param.Direction = ParameterDirection.Output;  
da.SelectCommand.Parameters.Add(param);
```

- 執行

```
objDataReader = da.SelectCommand.ExecuteReader();
```

- 讀取 output 參數的參數值

```
i = da.SelectCommand.Parameters("@ItemCount").Value;
```

# 呼叫預儲程序 (Stored Procedures)

- **Command 物件的各屬性值：**
  - CommandType = CommandType.StoredProcedure;
  - CommandText = 預儲程序的名稱
  - 利用 Parameters 備妥必要的參數

```
SqlDataAdapter daCategory = new SqlDataAdapter();  
daCategory.SelectCommand = new SqlCommand();  
daCategory.SelectCommand.Connection = conn;  
daCategory.SelectCommand.CommandText = "ProductCategoryList";  
daCategory.SelectCommand.CommandType = CommandType.StoredProcedure;
```



# 異動處理 (Transaction)

- 1 建立並開啟資料庫連線
- 2 呼叫連線物件的 `BeginTransaction()` 以取得 `Transaction` 物件
- 3 建立各個 `Command` 並且指定 `Transaction` 屬性為步驟 2 的那個 `Transaction` 物件
- 4 執行各個 `command` 物件的 `ExecuteXXX()` 方法
- 5 呼叫 `Transaction` 的 `Commit()` 或 `RollBack()` 方法進行確認或取消異動處理

# 異動的隔離等級 (Isolation Level)

隔離等級	Dirty reads?	Non-repeatable reads?	Phantom reads?
ReadUncommitted	Yes	Yes	Yes
ReadCommitted	No	Yes	Yes
RepeatableRead	No	No	Yes
Serializable	No	No	No

# 單元複習與討論



# 筆記

