

SQL advance

結合與子查詢

- 1 查詢多個資料表的資料
- 2 聯集
- 3 交集與差集
- 4 次查詢

# 1 查詢多個資料表的資料

- 關聯式資料庫中有一個可以將多個資料表組合起來以選取我們需要的資料的方法，稱之為**結合(Join)**；連結；聯結；合併)。
- 我們可以利用結合多個資料表來創建出一個新的資料表，而結合時會利用關聯欄位來連結資料表，也就是利用具有相同定義域的欄位來組合資料表。

# NATURAL JOIN

- 所謂NATURAL JOIN(自然結合)就是利用具**相同名稱**之特定的欄位來結合資料表，它可以避免結合後的欄位數量過多。最簡單的NATURAL JOIN之SQL語法如下：
  - `SELECT *`  
`FROM 資料表一 NATURAL JOIN 資料表二`

# NATURAL JOIN(續)

- 例如我們可以使用NATURAL JOIN來連結訂單主檔(Orders)與訂單明細檔(OrderDetails)如下：
  - **SELECT \***  
**FROM Orders NATURAL JOIN OrderDetails;**
- 以下為執行時的結果：

## NATURAL JOIN(續)

- 例如我們想限定訂單編號(OrderID)小於等於10250的資料，則可下達如下的指令：
  - **SELECT \***  
**FROM Orders NATURAL JOIN OrderDetails**  
**WHERE OrderID <= 10250**

## NATURAL JOIN(續)

- 對於NATURAL JOIN而言，其整個SQL語法如下：
  - **【NATURAL JOIN】**  
SELECT 表示式一 AS 別名一, 表示式二 AS 別名二, ...  
**FROM** 資料表一 **NATURAL JOIN** 資料表二  
WHERE 條件表示式  
GROUP BY 群組名稱一, 群組名稱二, ...  
HAVING 群組或聚合的篩選條件  
ORDER BY 排序欄位名稱一 [ASC|DESC],  
          排序欄位名稱二 [ASC|DESC], ...

# CROSS JOIN

- **交叉結合 (Cross Join)** 可以用來取得二個資料表的資料錄與資料錄的乘積，即關聯式代數裡的卡笛生乘積 (Cartesian Product)。
- 例如部門檔的資料有4筆，而員工檔的資料有15筆，則Cross Join的結果為  $4 * 15 = 60$  筆。
- 簡單的CROSS JOIN的語法：
  - **【CROSS JOIN】**  
SELECT \*  
FROM 資料表一 **CROSS JOIN** 資料表二或使用逗點分隔資料表名稱，而寫成：
  - **【CROSS JOIN】**  
SELECT \*  
FROM 資料表一, 資料表二



## CROSS JOIN(續)

- 例如我們想要列出所有的部門與員工的資料列組合，則可下達如下的指令：
  - **SELECT \***  
**FROM Departments CROSS JOIN Employees;**
- 或者
  - **SELECT \***  
**FROM Departments, Employees;**
- 而這個指令的查詢結果會傳回 $4*15=60$ 列。

## CROSS JOIN(續)

- 對SELECT查詢而言，其完整的CROSS JOIN語法如下：
  - 【SELECT指令格式】  
SELECT 表示式一 AS 別名一, 表示式二 AS 別名二, ...  
FROM 資料表來源名稱一, 資料表來源名稱二, ...  
WHERE 條件表示式  
GROUP BY 群組名稱一, 群組名稱二, ...  
HAVING 群組或聚合的篩選條件  
ORDER BY 排序欄位名稱一 [ASC|DESC],  
          排序欄位名稱二 [ASC|DESC], ...

注意：使用cross join時一般皆會有WHERE子句用於設定關聯條件或限制資料輸出量，若無WHERE子句則可能將造成龐大的資料輸出而造成系統無謂的負荷。

# Self-Join

- 這個查詢命令以Employees資料表和自身進行Join運算，所以Employees資料表同時扮演了兩個角色，因此必須使用別名的方式來表示其不同的角色。
- 查詢和員工王明亮同部門的其他員工的員工編號和姓名，則可下達如下的指令：

```
SELECT D.DepartmentID, D.EmployeeID, D.Name  
FROM Employees D, Employees E  
WHERE D.DepartmentID = E.DepartmentID  
AND E.Name = '王明亮'
```

- 自身連結會把王明亮的資料和其自身比較，造成自己等自己的狀況，因而必須加入一條限制式，以將王明亮本身的資料排除，即下如下的指令：

```
SELECT D.DepartmentID, D.EmployeeID, D.Name  
FROM Employees D, Employees E  
WHERE D.DepartmentID = E.DepartmentID  
AND E.Name = '王明亮'  
AND D.EmployeeID <> E.EmployeeID
```

# 作業:



- 11. 請查詢薪資大於Linda (Employees.EngName)的所有員工之資料。

## 4.3 聯集

- 聯集(Union)可用來將兩個查詢的結果集組合起來成為一個結果集，即集合裡的 $A \cup B$ 。UNION的語法如下：

**【UNION】**

SELECT 查詢指令一

UNION

SELECT 查詢指令二

- 其中兩個查詢指令的欄位數必須一樣，而且資料型態也必須能夠相容。

## 聯集(續)

- 例如組合客戶的公司名稱(CustomerID <= 'ANTON')  
和供應商的公司名稱(SupplierID <=3)：

```
SELECT '客戶' AS '類別', CompanyName
```

```
FROM Customers
```

```
WHERE CustomerID <= 'ANTON'
```

```
UNION
```

```
SELECT '供應商' AS '類別', CompanyName
```

```
FROM Suppliers
```

```
WHERE SupplierID <=3
```



## 聯集(續)

- 假如要聯集的資料超過兩個SQL指令，則使用格式樣式如下：

**【UNION】**

SELECT 查詢指令一

UNION

SELECT 查詢指令二

UNION

SELECT 查詢指令三

## 聯集(續)

- 使用UNION時會排除重複的資料。例如若SQL指令一為：  
SELECT EmployeeID, Name  
FROM Employees  
WHERE DepartmentID = 10
- 其執行的結果有3筆資料，為：

## 聯集(續)

- 而SQL指令二為：  
    SELECT EmployeeID, Name  
    FROM Employees  
    WHERE DepartmentID IN (10, 30)
- 其執行的結果有6筆資料，為：

## 聯集(續)

- 而當執行UNION時，即如下的指令：  
SELECT EmployeeID, Name  
FROM Employees  
WHERE DepartmentID = 10  
**UNION**  
SELECT EmployeeID, Name  
FROM Employees  
WHERE DepartmentID IN (10, 30)

## 聯集(續)

- 由於UNION會除去重複的資料，此時執行的結果有6筆資料，為：

## 聯集(續)

- 假如不想要除去重複的資料，則於UNION之後加入ALL字眼即可，即：

```
SELECT EmployeeID, Name
FROM Employees
WHERE DepartmentID = 10
UNION ALL
SELECT EmployeeID, Name
FROM Employees
WHERE DepartmentID IN (10, 30)
```

# 作業:

# homework

請用UNION聯集查詢

(一)查詢女性 (TitleOfCourtesy = '小姐')員工之最高薪資MAX(Salary)之員工資料

(二)查詢男性(TitleOfCourtesy = '先生')員工之最高薪資之 員工資料

EmployeeID, DepartmentID, Name, EngName, Salary, Commission, Title

# 交集與差集

- 交集是從兩個查詢結果集中取出共同存在的資料列作為交集的結果集，即集合的 $A \cap B$ ，其語法類似UNION，為如下：

SELECT 查詢指令一

**INTERSECT**

SELECT 查詢指令二



## 交集與差集(續)

- 而差集是從兩個查詢結果集中取出存在於第一個查詢但不存在於第二個查詢結果集的資料列作為差集的結果集，即集合的 $A - B$ ，其語法也類似UNION，為如下：

SELECT 查詢指令一

EXCEPT 或 MINUS

SELECT 查詢指令二

# 次查詢

- 使用SQL指令時，我們可以在一個SELECT查詢指令裡再嵌入另外一個查詢指令，換言之，我們可以將某一個查詢的結果拿來作為另外一個查詢的條件，這種查詢稱為次查詢或子查詢(Sub-Query)。
- 對於次查詢(一個SELECT指令)而言，其可能出現的時機有：
  - 位於另一個SELECT的WHERE子句、HAVING子句中。
  - 位於INSERT(新增)、UPDATE(修改)或DELETE(刪除)的子句中。
  - 在其他的次查詢之內，以形成階層式的架構。

## 次查詢(續)

- 次查詢出現於WHERE子句比較運算子 = 時的語法格式：

【次查詢】

SELECT 欄位列表

FROM 資料表列表

WHERE 表示式 =

(

SELECT 指令

)

## 次查詢(續)

- 例如我們想查詢產品名稱(ProductName)為 "牛奶"的  
訂單明細資料，但是我們並不清楚其產品編號。則  
我們可以下達如下的指令：

```
SELECT *  
FROM OrderDetails  
WHERE ProductID =  
(  
  SELECT ProductID  
  FROM Products  
  WHERE ProductName = '牛奶'  
)
```

## 次查詢(續)

- 這個查詢指令的運作方式為先執行內層的SELECT指令：

```
SELECT ProductID
```

```
FROM products
```

```
WHERE ProductName = '牛奶'
```

- 先取得牛奶的產品編號ProductID(1002)，然後將內層查詢到的ProductID(1002)資料傳給外層的WHERE子句，最後才執行外層的SELECT查詢。

## 次查詢(續)－多欄位

- 次查詢中的欄位的值，也可以是多個欄位。此時主查詢的欄位用括號括起來，而欄位數量和欄位屬性必需和子查詢的選取欄位數量一樣，且可以匹配。例如：查詢和郭國城同部門且同職稱的員工

```
SELECT EmployeeID, Name
FROM employees
WHERE (DepartmentID, Title) =
(
  SELECT DepartmentID, Title
  FROM employees
  WHERE Name = '郭國城'
)
```

# 次查詢(續)

- 依據外層**WHERE**子句所使用的組成元素，次查詢可分成三種類型：
  1. 傳回零或多筆資料的次查詢；用於**IN**、**ANY**或**ALL**之後。
  2. 只傳回單一筆資料的次查詢；用在一般的比較(如：**=**、**>=**…)運算子。
  3. 只用來測試存在性的次查詢；用在**EXISTS**之後。

# 傳回零或多筆資料的次查詢

- 使用IN(在...之中)或NOT IN(不在...之中)的次查詢語法如下：

【次查詢】

SELECT 欄位列表

FROM 資料表列表

WHERE 表示式 [NOT] IN

(

SELECT 指令

)



## 傳回零或多筆資料的次查詢(續)

- 例如我們想查詢訂單編號小於等於 10250之出貨產品的貨品編號與貨品名稱，則可下達如下的指令：

```
SELECT ProductID, ProductName
FROM products
WHERE ProductID IN
(
  SELECT ProductID
  FROM orderdetails
  WHERE OrderID <= 10250
)
```

## 傳回零或多筆資料的次查詢(續)

- 例如我們想查詢訂單編號  $\leq 10260$  且產品名稱含“起司”字眼的產品訂單明細資料，並將資料依產品編號排序，則可下達如下的指令：

```
SELECT *  
FROM orderdetails  
WHERE OrderID  $\leq$  10260  
AND ProductID IN  
(  
  SELECT ProductID  
  FROM products  
  WHERE ProductName LIKE '%起司%'  
)  
ORDER BY ProductID
```

## 傳回零或多筆資料的次查詢(續)

- 傳回零或多筆資料的次查詢的另一個使用時機為出現於**ANY**或**ALL**所修飾的比較運算子。此時其SQL指令格式為：

### 【次查詢】

SELECT 欄位列表

FROM 資料表列表

WHERE 表示式 比較運算子 ANY | ALL

(

SELECT 指令

)

# ALL與ANY的意義列表

ALL	結果	ANY	結果
> ALL (1, 2, 3)	同義於 > MAX 3	> ANY (1, 2, 3)	同義於 > MIN 1
< ALL (1, 2, 3)	同義於 < MIN 1	< ANY (1, 2, 3)	同義於 < MAX 3
= ALL (1, 2, 3)	=1 AND =2 AND =3 (同時成立)	= ANY (1, 2, 3)	=1 OR =2 OR =3 (任何一個成立)(同義於 IN)
<> ALL(1, 2, 3)	<>1 AND <>2 AND <>3(同義於 NOT IN)	<> ANY(1, 2, 3)	<>1 OR <>2 OR <>3

## 傳回零或多筆資料的次查詢(續)

- 例如我們想要查員工薪資大於任一個(ANY)職稱為 "業務"的非 "業務" 員工之員工編號、姓名、職稱及薪資。則可下達如下的指令：

```
SELECT EmployeeID, Name, Title, Salary
FROM employees
WHERE Title <> '業務'
AND Salary > ANY
(
  SELECT Salary
  FROM employees
  WHERE Title = '業務'
)
```

## 傳回零或多筆資料的次查詢(續)

- 而這個SQL指令事實上是同義於(比MIN大即可)：

```
SELECT EmployeeID, Name, Title, Salary
```

```
FROM employees
```

```
WHERE Title <> '業務'
```

```
AND Salary >
```

```
(
```

```
SELECT MIN(Salary)
```

```
FROM employees
```

```
WHERE Title = '業務'
```

```
)
```

## 傳回零或多筆資料的次查詢(續)

- 例如我們想要查詢員工薪資高於部門10的所有(ALL)員工之薪資的員工編號、姓名、職稱及薪資。則可下達如下的指令：

```
SELECT EmployeeID, Name, Title, Salary
FROM employees
WHERE Salary > ALL
(
  SELECT Salary
  FROM employees
  WHERE DepartmentID=10
)
```

# 傳回零或多筆資料的次查詢(續)

- 而這個指令事實上是同義於(比MAX大即可)：

```
SELECT EmployeeID, Name, Title, Salary  
FROM employees  
WHERE Salary >
```

(

```
SELECT MAX(Salary)  
FROM employees  
WHERE DepartmentID=10
```

)



## 傳回零或多筆資料的次查詢(續)

- 例如我們想要查出所有員工的任職部門不同於所有(<> ALL)員工編號為1001,1010等人的員工編號、姓名、部門編號及部門名稱，則可下達如下的指令：

```
SELECT EmployeeID, Name, DepartmentID
FROM employees
WHERE DepartmentID <> ALL
(
  SELECT DepartmentID
  FROM employees
  WHERE EmployeeID IN (1001, 1010)
)
```

# 測試存在性的次查詢

- 當次查詢是位於**WHERE**子句中的**EXISTS**關鍵字之後時，該次查詢的功能就變成存在性的檢查。**EXISTS**是用於檢查是否有資料符合次查詢的限制條件，而**NOT EXISTS**則是**EXISTS**的相反動作。使用**EXISTS**之次查詢的SQL指令格式為：

【次查詢】

```
SELECT 欄位列表  
FROM 資料表列表  
WHERE [NOT] EXISTS  
(  
  SELECT 指令  
)
```

## 測試存在性的次查詢(續)

- 例如我們想要查詢從 '1996-07-01' 至 '1996-07-15' 這段時間內曾下過訂單的客戶編號及公司名稱，則可下達如下的指令：

```
SELECT CustomerID, CompanyName
```

```
FROM customers
```

```
WHERE EXISTS
```

```
(
```

```
SELECT *
```

```
FROM orders
```

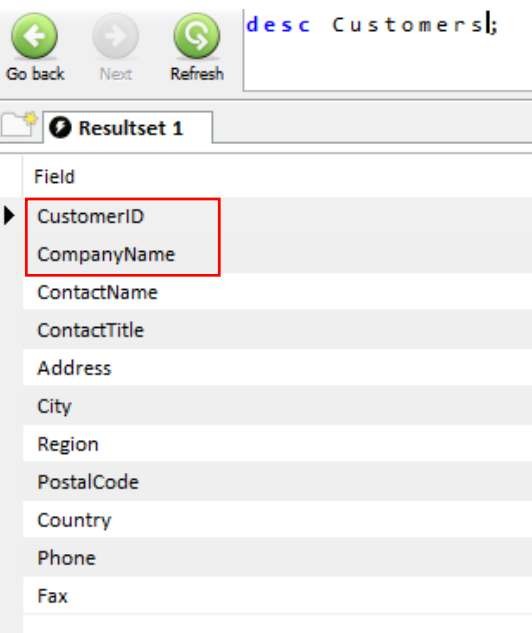
```
WHERE orders.CustomerID = customers.CustomerID
```

```
AND (OrderDate BETWEEN '1996-07-01' AND '1996-07-15')
```

```
)
```

# 多重巢狀的次查詢

- 在次查詢中可以包含另一個次查詢而形成一個多重巢狀的次查詢。
- 例如我們想要查1996年7月份有訂購產品編號1002(牛奶)的客戶編號及客戶名稱，則可下達如下的指令：



Go back Next Refresh desc Customers;

Resultset 1

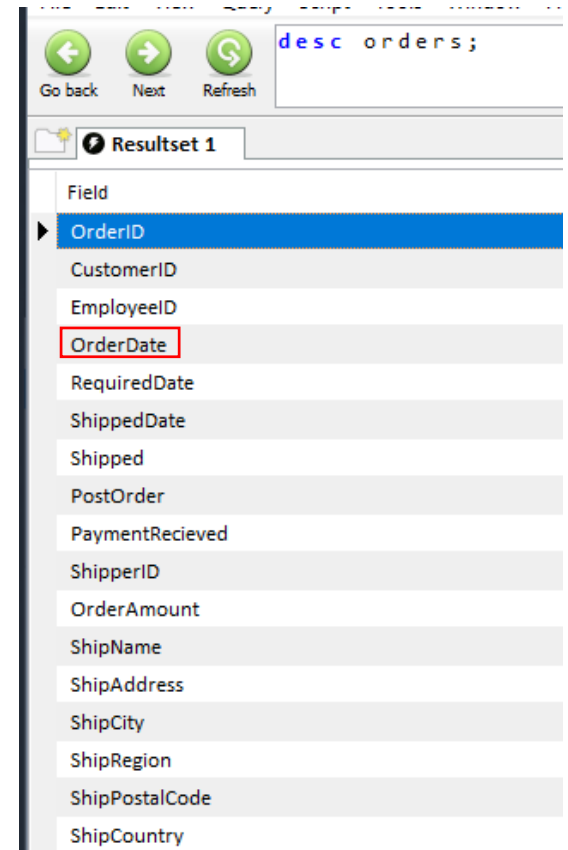
Field
CustomerID
CompanyName
ContactName
ContactTitle
Address
City
Region
PostalCode
Country
Phone
Fax



Go back Next Refresh desc orderdetails;

Resultset 1

Field	Type
OrderID	int(11)
ProductID	int(11)
UnitPrice	double
Quantity	smallint(6)
Discount	double



Go back Next Refresh desc orders;

Resultset 1

Field
OrderID
CustomerID
EmployeeID
OrderDate
RequiredDate
ShippedDate
Shipped
PostOrder
PaymentRecieved
ShipperID
OrderAmount
ShipName
ShipAddress
ShipCity
ShipRegion
ShipPostalCode
ShipCountry

## 多重巢狀的次查詢(續)

```
SELECT CustomerID, CompanyName
FROM customers
WHERE CustomerID IN
(
  SELECT CustomerID
  FROM orders
  WHERE (OrderDate BETWEEN '1996-07-01' AND '1996-07-31')
  AND EXISTS
  (
    SELECT *
    FROM orderdetails
    WHERE OrderID = Orders.OrderID
    AND ProductID = 1002
  )
)
```

# 作業:

# homework

- 請查詢1996年7月份的訂單的訂貨內容

Hint.

1. YEAR(Orders.OrderDate) = 1996

2. SELECT OrderDetails.OrderID, OrderDetails.ProductID,  
OrderDetails.UnitPrice, OrderDetails.Quantity, OrderDetails.Discount

From ...