第 8 章

SQL-99：綱要定義、基本限制與查詢

學習重點

- CREATE TABLE、DROP TABLE、ALTER TABLE、CREATE SCHEMA
- 參考完整性限制的選項
- SQL2與SQL-99的新增資料型態
- SQL中的基本查詢語法
- SELECT-FROM-WHERE結構
- 簡單的SQL查詢
- 別名、* 和DISTINCT的用法
- 未指定的WHERE子句
- 集合運算
- 巢狀查詢與相互關聯的巢狀查詢

學習重點

- EXISTS函數
- 明確指定的集合
- SQL查詢中的NULL
- SQL2的合併表格功能
- 聚合函數、分組與HAVING子句
- 子字串比對與算術運算
- ORDER BY
- SQL查詢語法總整理
- INSERT、DELETE、UPDATE

變更資料定義、限制或綱要

- 用來建立 (CREATE)、刪除 (DROP) 和修改 (ALTER) 資料庫中表格 (關聯) 的描述

CREATE TABLE

- 用來建立一個新的關聯，其方式為指定關聯的名稱、每個屬性即它們的資料型態 (INTEGER、FLOAT、DECIMAL(i,j)、CHAR(n)、VARCHAR(n))
- 在屬性上還可能會指定限制 (如NOT NULL)

```
CREATE TABLE   DEPARTMENT
      (          DNAME VARCHAR(10)    NOT NULL,
                 DNUMBER        INTEGER         NOT NULL,
                 MGRSSN         CHAR(9),
                 MGRSTARTDATE           CHAR(9)  );
```


CREATE TABLE

- 在SQL2可使用CREATE TABLE命令來指定主鍵屬性、替代鍵和參考完整性限制 (外來鍵)
- 鍵值屬性是使用PRIMARY KEY和UNIQUE子句來指定

```
CREATE TABLE   DEPT
(    DNAMEVARCHAR(10) NOT NULL,
     DNUMBER       INTEGER         NOT NULL,
     MGRSSN        CHAR(9),
     MGRSTARTDATE        CHAR(9),
     PRIMARY KEY (DNUMBER),
     UNIQUE (DNAME),
     FOREIGN KEY (MGRSSN) REFERENCES EMP  );
```

DROP TABLE

- 用來刪除綱要裡的關聯 (基底表格) 及其定義
- 之後在任何查詢或其他命令中就不能再用到這個關聯，因為它的定義資訊已經不存在了
- 範例：

DROP TABLE DEPENDENT;

ALTER TABLE

- 用來變更基底關聯的定義
- 假如是新增一個屬性，則在剛執行完此命令時，關聯中所有值組的這個新屬性其值都是NULL，因此這時不允許有NOT NULL限制
- 範例：

ALTER TABLE EMPLOYEE ADD JOB VARCHAR(12);

- 之後資料庫使用者還是必須替每筆EMPLOYEE值組輸入新的屬性JOB的值。這可以使用UPDATE命令來做

SQL2與SQL-99的新增功能

- **CREATE SCHEMA**
- **REFERENTIAL INTEGRITY OPTIONS**

CREATE SCHEMA

- 指定名稱建立一個新的資料庫綱要

參考完整性限制的選項

- 可針對參考完整性限制 (外來鍵) 指定RESTRICT、CASCADE、SET NULL或SET DEFAULT

```
CREATE TABLE   DEPT
(     DNAME          VARCHAR(10)  NOT NULL,
      DNUMBER        INTEGER          NOT NULL,
      MGRSSN         CHAR(9),
      MGRSTARTDATE       CHAR(9),
      PRIMARY KEY (DNUMBER),
      UNIQUE (DNAME),
      FOREIGN KEY (MGRSSN) REFERENCES EMP
ON DELETE SET DEFAULT ON UPDATE CASCADE  );
```


參考完整性限制的選項 (續)

```
CREATE TABLE   EMP
    (         ENAME          VARCHAR(30)  NOT NULL,
              ESSN   CHAR(9),
              BDATE DATE,
              DNO    INTEGER  DEFAULT 1,
              SUPERSSN      CHAR(9),
              PRIMARY KEY (ESSN),
              FOREIGN KEY (DNO) REFERENCES DEPT
           ON DELETE SET DEFAULT ON UPDATE CASCADE,
              FOREIGN KEY (SUPERSSN) REFERENCES EMP
           ON DELETE SET NULL ON UPDATE CASCADE  );
```

SQL2與SQL-99的新增資料型態

包括DATE、TIME和TIMESTAMP資料型態

- **DATE：**
  - 由年-月-日所組成，格式為yyyy-mm-dd
- **TIME：**
  - 由時:分:秒所組成，格式為hh:mm:ss
- **TIME(i)：**
  - 由時:分:秒加上代表幾分之一秒的數字 i 所組成
  - 格式為hh:mm:ss:ii...i

SQL2與SQL-99的新增資料型態

- **TIMESTAMP (時間戳記)：**
  - 由DATE和TIME元件所組成
- **INTERVAL (期間)：**
  - 指定一個相對值，而不是絕對值
  - 可能是YEAR/MONTH期間或DAY/TIME期間
  - 當它相加或相減另一個絕對值，結果可能是正數或負數，結果也會是絕對值

SQL中的基本查詢語法

- SQL有個很重要的基本敘述，可以讓我們從資料庫中擷取資訊，也就是SELECT敘述
- 它與第6章所討論的關聯式代數的SELECT運算是不一樣的
- SQL與關聯式模型之間有個很重要的差異：SQL允許表格 (即關聯) 中存在兩筆或多筆所有屬性值完全相同的值組
- 因此，通常SQL表格並非一個值組的集合，因為集合不允許有重複的值組。SQL表格是值組的多重集合 (multiset，或稱為bag)
- 不過SQL關聯如果有指定PRIMARY KEY或UNIQUE屬性，或是在SELECT命令中加上DISTINCT選項，此時的SQL關聯就必須是值組的集合

SELECT-FROM-WHERE結構

- 基本的SELECT敘述，有時也稱為SELECT-FROM-WHERE區塊 (*block)*

SELECT <attribute list>
FROM <table list>
WHERE <condition>

  - <attribute list> 是屬性名稱的列表，在查詢時需要參考這些屬性的值
  - <table list> 列出處裡查詢時會用到的關聯
  - <condition> 是條件（布林）運算式，用來識別查詢時所要擷取的值組

範例關聯式資料庫綱要 (圖5.5)

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

DEPT_LOCATIONS

DNUMBER	DLOCATION

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

WORKS_ON

ESSN	PNO	HOURS

DEPENDENT

ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP

範例資料庫的某個狀態 (圖5.6)

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-07-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

DEPT_LOCATIONS

DNUMBER	DLOCATION
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

WORKS_ON

ESSN	PNO	HOURS
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	null

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP
333445555	Alice	F	1986-04-05	DAUGHTER
333445555	Theodore	M	1983-10-25	SON
333445555	Joy	F	1958-05-03	SPOUSE
987654321	Abner	M	1942-02-28	SPOUSE
123456789	Michael	M	1988-01-04	SON
123456789	Alice	F	1988-12-30	DAUGHTER
123456789	Elizabeth	F	1967-05-05	SPOUSE

簡單的SQL查詢

- 基本的SQL查詢是對應關聯式代數中的SELECT、PROJECT和JOIN運算
- 接下來所有的範例都是使用COMPANY資料庫
- 每個簡單查詢範例是針對一個關聯
- 查詢範例0：擷取名叫 'John B. Smith' 員工的生日與住址

```
Q0: SELECT BDATE, ADDRESS
    FROM          EMPLOYEE
    WHERE  FNAME='John' AND MINIT='B'
    AND           LNAME='Smith'
```

  - 類似於關聯式代數的SELECT-PROJECT運算配對，其中SELECT子句負責指定投影屬性，而WHERE子句則負責指定選擇條件
  - 不過，查詢的結果可能會有重複的值組

簡單的SQL查詢 (續)

- 查詢範例1：擷取所有在 'Research' 部門工作的員工的姓名與住址

```
Q1:SELECT     FNAME, LNAME, ADDRESS
   FROM EMPLOYEE, DEPARTMENT
   WHERE      DNAME='Research' AND
   DNUMBER=DNO
```

  - 類似關聯式代數運算中的SELECT-PROJECT-JOIN
  - (DNAME='Research') 是選擇條件 (相當於關聯式代數的SELECT運算)
  - (DNUMBER=DNO) 則是合併條件 (相當於在關聯式代數的JOIN運算)

簡單的SQL查詢 (續)

- 查詢範例2：列出所有位在 'Stafford' 地點的計畫其計畫編號、控管部門編號，以及部門經理的姓氏、住址和生日

```
Q2: SELECT    PNUMBER, DNUM, LNAME, BDATE,
      ADDRESS
    FROM      PROJECT, DEPARTMENT, EMPLOYEE
    WHERE     DNUM=DNUMBER AND MGRSSN=SSN
          AND       PLOCATION='Stafford'
```

  - 在Q2裡有兩個合併條件
  - 合併條件DNUM=DNUMBER使得計劃與其控管部門產生關聯
  - 合併條件MGRSSN=SSN則讓控管部門與管理此部門的員工產生關聯

別名、* 和DISTINCT、空的WHERE子句

- 在SQL中，只要屬性是屬於不同的關聯，就可以讓兩個或多個屬性使用同樣的名稱
- 若查詢會參考到兩個或多個同名的屬性，就必須用關聯名稱來修飾 (qualify) 屬性名稱，做法是將關聯名稱放在屬性名稱之前，並用英文的句點 (.) 來分隔

範例：

- EMPLOYEE.LNAME、DEPARTMENT.DNAME

別名

- 有些查詢需要對同一個關聯參考兩次
- 在這類情況要對關聯名稱 *aliases* are given to the relation name
- 查詢範例8：擷取每一位員工的姓名與其直屬上司的姓名

```
Q8:  SELECT     E.FNAME, E.LNAME, S.FNAME,
                        S.LNAME
     FROM       EMPLOYEE E S
     WHERE      E.SUPERSSN=S.SSN
```

  - 在Q8中的替代關聯名稱E和S被稱作EMPLOYEE關聯的別名 (*alias*) 或值組變數 (*tuple*)

別名 (續)

- 我們可以將E和S想像成EMPLOYEE的兩份不同的副本；是代表扮演部屬角色的員工，而S則是代表扮演上司角色的員工
- 別名可以用在任何SQL查詢中
- 也可以使用AS關鍵字來指定別名

```
Q8: SELECT    E.FNAME, E.LNAME, S.FNAME,
                  S.LNAME
      FROM    EMPLOYEE AS E, EMPLOYEE AS S
     WHERE    E.SUPERSSN=S.SSN
```

未指定的WHERE子句

- 假如沒有WHERE的子句，代表沒有選擇條件，因此FROM子句裡所指定關聯的所有值組都會被選取
- 這就相當於條件是WHERE TRUE
- 查詢範例9：在資料庫中選擇所有員工的SSN資料

Q9: SELECT SSN
FROM EMPLOYEE

- 假如在FROM子句中指定一個以上的關聯，而且沒有WHERE子句，則表示結果等於這些關聯的CROSS PRODUCT，也就是所有可能的值組組合都會被選取

未指定的WHERE子句 (續)

- 查詢範例10：選擇EMPLOYEE SSN與DEPARTMENT DNAME的所有組合

Q10:SELECT SSN, DNAME
FROM EMPLOYEE, DEPARTMENT

  - 在WHERE子句中確實指定每一個選擇條件與合併條件是非常重要的，否則可能會產生不正確或者是相當龐大的關聯

* 的用法

- 在SQL中，假如要擷取所選值組的所有屬性值，只需要用一個星號 (*) 即可，這個星號就代表所有的屬性，例如：

```
Q1C:    SELECT      *
        FROM  EMPLOYEE
        WHERE       DNO=5

Q1D:    SELECT      *
        FROM  EMPLOYEE, DEPARTMENT
        WHERE       DNAME='Research' AND
                    DNO=DNUMBER
```


DISTINCT的用法

- SQL通常不會將關聯視為集合，因此可以出現重複的值組
- 為了消除查詢結果中的重複值組，可使用關鍵字 **DISTINCT**
- 例如Q11的結果可能會有重複的SALARY值，而Q11A則不會有任何重複值

```
Q11:SELECT      SALARY
        FROM  EMPLOYEE

Q11A:   SELECT      DISTINCT SALARY
        FROM  EMPLOYEE
```

集合運算

- SQL已經直接整合關聯式代數的某些集合運算
- 通常都會有聯集運算 (UNION)，有些SQL 版本還會有差集 (MINUS) 和交集(INTERSECT) 運算
- 這些集合運算所產生的關聯是值組的集合；也就是說，重複的值組會在結果中被除去
- 集合運算只能應用在聯集相容的關聯上，所以必須先確定要運算的兩個關聯具有相同的屬性，而且這些屬性出現在兩個關聯的順序也相同

集合運算 (續)

- 查詢範例4：列出姓氏為 'Smith' 的員工所參與的所有計畫，不論該員工是計畫的工作人員或是管理此計畫的部門經理

```
Q4:  (SELECT  PNAME
     FROM          PROJECT, DEPARTMENT, EMPLOYEE
     WHERE DNUM=DNUMBER AND MGRSSN=SSN     AND
     LNAME='Smith')
     UNION         (SELECT  PNAME
     FROM          PROJECT, WORKS_ON, EMPLOYEE
     WHERE PNUMBER=PNO AND ESSN=SSN AND
               LNAME='Smith')
```

巢狀查詢

- 所謂的巢狀查詢 (nested query) 是指在一個查詢的WHERE子句內，含有完整的「SELECT-FROM-WHERE」區塊。此時這個外部的WHERE查詢被稱為外部查詢 (outer query)
- 很多之前的查詢範例都可以使用巢狀查詢來改寫
- 查詢範例1：擷取所有在 'Research' 部門工作的員工的姓名與住址

```
Q1:  SELECT        FNAME, LNAME, ADDRESS
     FROM  EMPLOYEE
     WHERE         DNO IN  (SELECT  DNUMBER
                   FROM  DEPARTMENT
                   WHERE         DNAME='Research' )
```


巢狀查詢 (續)

- 由巢狀查詢先選出 'Research' 部門的編號
- 外部查詢來選擇其DNO值屬於巢狀查詢結果中的EMPLOYEE值組
- 這裡的比較運算子IN是v值與由V值所組成的集合 (或多重集合)，若v隸屬於V則結果為TRUE
- 通常可以使用好幾層巢狀查詢
- 假如屬性沒有明確指定是哪個關聯，那麼就以最內層 (innermost) 的查詢所宣告的關聯為準
- 此例的巢狀查詢與外部查詢沒有關聯

相互關聯的巢狀查詢

- 假如巢狀查詢中WHERE子句中的條件，會參考到宣告在外部查詢的關聯裡的某些屬性，就稱這兩個查詢是相互關聯的(correlated)
- 相互關聯的巢狀查詢其結果與外部查詢關聯的每個值組 (或值組組合) 不同
- 查詢範例16：擷取眷屬的名字和性別與員工本人相同的員工姓名

```
Q16: SELECT   E.FNAME, E.LNAME
     FROM     EMPLOYEE AS E
     WHERE    E.SSN IN (SELECT    ESSN
                        FROM      DEPENDENT
                        WHERE E.FNAME=DEPENDENT_NAME
                AND E.SEX=D.SEX);
```


相互關聯的巢狀查詢 (續)

- 在Q16中，巢狀查詢其結果與外部查詢關聯的結果值組不同
- 以巢狀「SELECT-FROM-WHERE」區塊所撰寫，而且使用「=」或「IN」比較運算子的查詢，一定可以改寫成單一區塊的查詢。例如Q16可以改寫成Q16A：

```
Q16A:   SELECT   E.FNAME, E.LNAME
        FROM     EMPLOYEE AS E, DEPENDENT AS D
        WHERE    E.SSN=D.ESSN
                 AND E.SEX=D.SEX
                 AND E.FNAME=D.DEPENDENT_NAME
```

EXISTS函數

- EXISTS函數，是用來檢查相互關聯的巢狀查詢的結果是否為空的 (沒有任何值組)
- 將查詢範例16用EXISTS改寫成Q16B如下：

```
Q16B:   SELECT  E.FNAME, E.LNAME
                FROM    EMPLOYEE AS E
                WHERE  EXISTS  (SELECT       *
                        FROM    DEPENDENT
                        WHERE    E.SSN=ESSN
                        AND E.SEX=SEX
                        AND E.FNAME=DEPENDENT_NAME
```


EXISTS函數 (續)

- 查詢範例6：擷取沒有眷屬的員工姓名

```
Q6:  SELECT         FNAME, LNAME
     FROM  EMPLOYEE
     WHERE          NOT EXISTS   (SELECT         *
                            FROM  DEPENDENT
                            WHERE SSN=ESSN)
```

  - 在Q6中，相互關聯的巢狀查詢會擷取所有與EMPLOYEE值組相關聯的DEPENDENT值組，如果不存在則此EMPLOYEE值組會被選取
  - 對於SQL的表達能力而言，EXISTS是必要的

明確指定的集合

- 也可以在WHERE子句中，使用明確數值的集合來取代巢狀查詢
- 查詢範例17：擷取所有在1、2或3號計畫工作的員工的社會安全號碼

```
Q13:    SELECT    DISTINCT ESSN
        FROM      WORKS_ON
        WHERE     PNO IN  (1, 2, 3)
```


SQL查詢中的NULL

- SQL可以檢查某個值是否為NULL (未知或無法取得或不適用)
- SQL是使用IS或IS NOT來比較NULL，因為SQL將每一個空值都視為與其他的空值不同，所以不應該比較是否相等
- 查詢範例18：擷取所有沒有上司的員工姓名

```
Q14:    SELECT    FNAME, LNAME
        FROMEMPLOYEE
        WHERE     SUPERSSN  IS  NULL
```

注意：在指定合併條件時，合併屬性為空值的值組不會被包含在結果內

SQL2的合併表格功能

- 可以在FROM子句中指定合併運算所得結果的表格
- 看起來和其他關聯一樣，不過它是合併運算的結果
- 使用者可以指定各種不同類型的合併運算(一般的“theta” JOIN、NATURAL JOIN、LEFT OUTER JOIN、RIGHT OUTER JOIN、CROSS JOIN等)

SQL2的合併表格功能 (續)

- 範例：

Q8: SELECTE.FNAME, E.LNAME, S.FNAME, S.LNAME
FROM EMPLOYEE E S
WHERE E.SUPERSSN=S.SSN

可以改寫成：

Q8: SELECTE.FNAME, E.LNAME, S.FNAME, S.LNAME
FROM (EMPLOYEE E LEFT OUTER JOIN EMPLOYEES
ON E.SUPERSSN=S.SSN)

SQL2的合併表格功能 (續)

```
Q1:  SELECT FNAME, LNAME, ADDRESS
     FROM   EMPLOYEE, DEPARTMENT
     WHERE DNAME='Research' AND
     DNUMBER=DNO
```

可以改寫成：

```
Q1A:SELECT FNAME, LNAME, ADDRESS
     FROM   (EMPLOYEE JOIN DEPARTMENT
             ON DNUMBER=DNO)
     WHERE DNAME='Research'
```

或者是：

```
Q1B:SELECT FNAME, LNAME, ADDRESS
     FROM   (EMPLOYEE NATURAL JOIN DEPARTMENT
             AS DEPT(DNAME, DNO, MSSN, MSDATE)
     WHERE DNAME='Research'
```


SQL2的合併表格功能 (續)

- 另一個範例：
  - Q2可以改寫如下。合併條件也可能是巢狀的；也就是說，要合併的表格本身就是一個已合併的表格

```
Q2A:    SELECT      PNUMBER, DNUM, LNAME,
                    BDATE, ADDRESS
        FROM (PROJECT JOIN
                DEPARTMENT ON
        DNUM=DNUMBER) JOIN
  EMPLOYEE ON
  MGRSSN=SSN) )
        WHERE       PLOCATION='Stafford'
```

聚合函數

- 包括**COUNT**、**SUM**、**MAX**、**MIN**和**AVG**
- 查詢範例19：計算出所有員工薪資的總和、最高薪資、最低薪資和平均薪資

```
Q19:    SELECT    MAX(SALARY),
                  MIN(SALARY),
                  AVG(SALARY)
        FROMEMPLOYEE
```

  - 有些SQL版本可能不允許在SELECT子句中有一個以上的函數

聚合函數 (續)

- 查詢範例20：計算出所有在 'Research' 部門工作的員工其薪資總和，以及此部門員工的最高薪資、最低薪資與平均薪資

```
Q20: SELECT    MAX(SALARY), MIN(SALARY),
               AVG(SALARY)
     FROMEMPLOYEE, DEPARTMENT
     WHERE     DNO=DNUMBER AND
               DNAME='Research'
```

聚合函數 (續)

- 查詢範例21和22：計算公司裡員工的總數 (Q21)，以及在 'Research' 部門工作的員工總數 (Q22)

```
Q21: SELECT          COUNT (*)
     FROM   EMPLOYEE

Q22: SELECT          COUNT (*)
     FROM   EMPLOYEE,      DEPARTMENT
            WHERE DNO=DNUMBER AND
                  DNAME='Research'
```


分組

- 在很多時候會需要將聚合函數應用在關聯中，根據某些屬性值分類的值組子群組上
- 每個子群組是由指定的群組化屬性 (grouping attribute) 分組而成，每一組的群組化屬性值是相同的
- 真對每個子群組分別使用函數
- SQL提供GROUP BY子句用來指定群組化屬性，它一定要出現在SELECT子句中

分組 (續)

- 查詢範例24：列出每個部門的編號，以及此部門的工作員工人數和平均薪資

```
Q24:SELECT      DNO, COUNT (*), AVG (SALARY)
    FROM  EMPLOYEE
    GROUP BY    DNO
```

  - 在Q24將EMPLOYEE值組分成數個群組，每個群組都有相同的群組化屬性DNO值
  - 而在每個值組群組中分別執行COUNT與AVG函數
  - SELECT子句只包含群組化屬性及在每個值組群組上執行的函數
  - 合併條件可與分組功能搭配使用

分組 (續)

- 查詢範例25：擷取每一個計畫的計畫編號、計畫名稱、以及在此計畫裡工作的員工人數

```
Q25:SELECT      PNUMBER, PNAME, COUNT (*)
            FROM  PROJECT, WORKS_ON
            WHERE       PNUMBER=PNO
            GROUP BY    PNUMBER, PNAME
```

  - 此例的分組與函數要等到兩個關聯合併之後才會執行

HAVING子句

- 有時候會只需要擷取滿足特定條件群組的函數值
- 此時可使用HAVING子句在群組 (而不是個別值組) 上指定選擇條件

HAVING子句 (續)

- 查詢範例26：對工作員工在兩人以上的每個計畫，擷取其計畫編號、計畫名稱以及計畫中的工作員工人數

```
Q26:   SELECT     PNUMBER, PNAME, COUNT (*)
       FROM       PROJECT, WORKS_ON
       WHERE      PNUMBER=PNO
       GROUP BY   PNUMBER, PNAME
       HAVING     COUNT (*) > 2
```

子字串比對

- LIKE比較運算子可用來比對子字串
- 部份字串的指定方式是藉由兩個保留字元：百分比「%」字元可取代任意數目的字元，底線「_」字元則可取代單一字元

子字串比對 (續)

- 查詢範例12：擷取所有住址在Houston, Texas的員工
  - 也就是說ADDRESS屬性的值必須包含子字串 'Houston,TX‘

```
Q12:    SELECT    FNAME, LNAME
        FROM      EMPLOYEE
        WHERE     ADDRESS LIKE
                  '%Houston,TX%’
```

子字串比對 (續)

- 查詢範例12A：擷取所有在1950年代出生的員工
  - 根據日期的格式，此例 '5' 必須是字串的第3個字元，因此BDATE值是 '_ _ 5 _ _ _ _ _ _ _ _'，這裡的底線字元可以是任何一個字元

```
Q12A:  SELECT    FNAME, LNAME
       FROM EMPLOYEE
       WHERE     BDATE LIKE '_ _ 5 _ _ _ _ _ _ _'
```


算術運算

- 在數值的資料或屬性上可使用一般的標準四則運算 (+、-、*、/)
- 查詢範例13：顯示出所有工作於 'ProductX' 計畫的員工加薪 10% 後的薪資結果

```
Q13: SELECT    FNAME, LNAME, 1.1*SALARY
     FROM EMPLOYEE, WORKS_ON, PROJECT
     WHERE     SSN=ESSN AND PNO=PNUMBER AND
               PNAME='ProductX'
```

ORDER BY

- ORDER BY子句是用來針對值組內的一或多個屬性值，將查詢結果的值組加以排序
- 查詢範例15：擷取員工與他們所工作計畫的清單，在清單中先針對部門號碼排序，每個部門內再依員工姓名的字母順序排序

```
Q15:    SELECT      DNAME, LNAME, FNAME, PNAME
        FROM        DEPARTMENT, EMPLOYEE,
                    WORKS_ON, PROJECT
        WHERE DNUMBER=DNO AND SSN=ESSN
                AND PNO=PNUMBER
        ORDER BY    DNAME, LNAME
```


ORDER BY (續)

- 預設的順序是遞增排序
- 使用關鍵字DESC可變成遞減排序，而關鍵字ASC則是用來更明確的指定遞增排序

SQL查詢語法總整理

- SQL的查詢最多可包含6個子句，但只有前兩個SELECT與FROM子句是必要的。子句是以下列的順序來指定：

```
SELECT      <attribute list>
FROM        <table list>
[WHERE      <condition>]
[GROUP BY <grouping attribute(s)>]
[HAVING     <group condition>]
[ORDER BY <attribute list>]
```


SQL查詢語法總整理 (續)

- SELECT子句中列出被擷取的屬性或函數
- FROM子句是指定所有在查詢時需要的關聯 (表格)，包括合併的關聯，但不包括巢狀查詢所需的關聯
- WHERE子句是指定從這些關聯裡選取值組的條件，同時視需要加入合併條件
- GROUP BY子句中指定群組化屬性
- HAVING子句所指定的是群組的選取條件
- ORDER BY用來指定查詢結果的顯示順序
- 理論上查詢最早執行的會是FROM子句，接著是WHERE子句，然後是GROUP BY和HAVING子句，最後是ORDER BY子句用來將查詢結果排序

SQL的修改命令

- 在SQL中有3個命令可以用來修改資料庫：INSERT、DELETE和UPDATE

INSERT

- 最簡單的INSERT形式是在關聯中加入一筆值組
- 在指定值組內欄位的值時，必須以CREATE TABLE命令指定的屬性順序來排列

INSERT (續)

- 範例：

U1: INSERT INTO EMPLOYEE
VALUES ('Richard','K','Marini', '653298653', '30-DEC-52',
'98 Oak Forest,Katy,TX', 'M', 37000,'987654321', 4)

- 另一種形式是允許使用者明確的指定對應到INSERT命令裡的數值的屬性名稱
- 可以有NULL或DEFAULT值的屬性則可以不指定資料值
- 範例：輸入一筆只指定FNAME、LNAME和SSN屬性的新EMPLOYEE值組

U1A: INSERT INTO EMPLOYEE (FNAME, LNAME, SSN)
VALUES ('Richard', 'Marini', '653298653')

INSERT (續)

- 注意：只有以DDL命令指定的限制，DBMS才會在變更資料庫時自動遵守
- INSERT命令有種變化形式，就是在一個查詢中插入多個值組到關聯

INSERT (續)

- 範例：假設要建立具有每個部門的名稱、員工人數與總薪資的暫存表格。首先使用U3A建立DEPTS_INFO表格，然後再使用U3B從資料庫中擷取摘要資訊

```
U3A:    CREATE TABLE  DEPTS_INFO
                (DEPT_NAME   VARCHAR(10),
                 NO_OF_EMPS INTEGER,
                 TOTAL_SAL    INTEGER);

U3B:    INSERT INTO   DEPTS_INFO (DEPT_NAME,
                      NO_OF_EMPS, TOTAL_SAL)
        SELECT        DNAME, COUNT (*), SUM (SALARY)
        FROM          DEPARTMENT, EMPLOYEE
        WHERE         DNUMBER=DNO
        GROUP BY      DNAME ;
```


INSERT (續)

- 注意：DEPTS_INFO表格可能不是最新的；也就是說，假如在執行過U3B之後才更新DEPARTMENT或EMPLOYEE關聯，則DEPTS_INFO內的資訊就會變成過時的資訊。因此最好是建立所謂的視界 (view)，才能讓表格的資料維持最新

DELETE

- 用來將值組從關聯中移除
- 包括WHERE子句來選取要被刪除的值組
- 一次只能從一個表格中刪除值組 (除非在參考完整性限制上有指定CASCADE)
- 假如沒有指定WHERE子句，則會將值組全部刪除，最後把這個表格當作一個空的表格留在資料庫中
- 被刪除的值組個數是根據符合WHERE子句條件的個數
- 一定會遵守參考完整性限制

DELETE (續)

- 範例：

```
U4A:    DELETE FROM     EMPLOYEE
        WHERE           LNAME='Brown'

U4B:    DELETE FROM     EMPLOYEE
        WHERE           SSN='123456789'

U4C:    DELETE FROM     EMPLOYEE
        WHERE           DNO  IN
                (SELECT         DNUMBER
                FROM  DEPARTMENT
                WHERE           DNAME='Research')

U4D:    DELETE FROM     EMPLOYEE
```

UPDATE

- 用來修改一或多個被選取值組的屬性值
- WHERE子句從關聯中選取要修改的值組
- SET子句是用來指定被修改的屬性和它的新數值
- 每個命令修改同一個關聯裡的值組
- 一定會遵守參考完整性限制

UPDATE (續)

- 範例：將10號計劃的位置與控管部門編號分別改成'Bellaire' 和5

```
U5:  UPDATE      PROJECT
     SET         PLOCATION = 'Bellaire', DNUM = 5
     WHERE       PNUMBER=10
```

UPDATE (續)

- 範例：將所有在 'Research' 部門工作的員工加薪10%

```
U6:  UPDATE          EMPLOYEE
     SET             SALARY = SALARY *1.1
     WHERE DNO  IN (SELECT        DNUMBER
                    FROM          DEPARTMENT
                    WHERE         DNAME='Research')
```

- 因為修改後的薪資值是依據每個值組原先的薪資值，所以需要參考SALARY屬性兩次
- U6中右邊的SALARY屬性的參考動作是參考在修改前的舊SALARY值
- 左邊的是參考在修改之後的新SALARY值

學習評量

- 請列出SQL所允許的合法的屬性資料型態。
- 請描述SQL查詢中的6個子句的語法？並分別說明這6個子句中分別可以指定什麼種類的元件？這6個子句中哪些是必要的？那些是選用的？
- 請說明在理論上SQL查詢的6個子句的執行順序。
- 請說明SQL在執行比較運算時如何處理NULL值。假如在SQL查詢中有用到聚合函數時，又是如何處理NULL值？如果在群組化屬性中有NULL時又是如何處理？

學習評量

- 以圖1.2的資料庫為例，其綱要如圖2.1。在此綱要中哪一個是必須保持的參考完整性限制？請撰寫適當的SQL DDL敘述來定義資料庫。
- 請以SQL語言來撰寫下列關於圖5.5資料庫的查詢，假設每個查詢都應用在圖5.6的資料庫上。請分別說明查詢的結果。
  - a. 對員工平均薪資超過 $30,000的每個部門，擷取其部門名稱與此部門的工作員工人數。
  - b. 假設想要擷取在每個部門工作的男性員工人數，而不是如習題 (a) 的全部員工人數。請問能夠用SQL來寫出這樣的查詢嗎？理由為何？