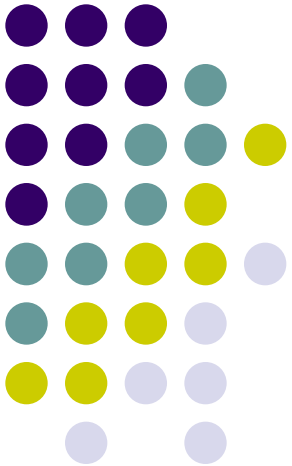
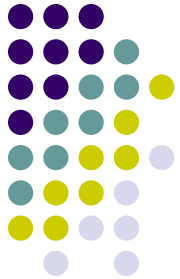
# 第 8 章

## Part I
## SQL 查詢



# SQL簡介



- 結構化查詢語言
  - SQL (Structured Query Language)
- 關連式資料庫系統的標準語言
- SQL功能包括
  - 資料定義、查詢與更新敘述
  - 既是DDL, 也是DML
  - 定義資料庫視界、安全性與權限、完整性及交易控制
  - 具備嵌入Java、COBOL或C/C++等語言的機制

# SELECT-FROM-WHERE結構

**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名稱
- <condition> 是條件（布林）運算式，用來識別查詢時所要擷取的值組

# 圖5.5
# 範例關聯式資料庫綱要

**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 |
|---|---|---|---|---|

# 圖5.6

# 範例資料庫的某個狀態

| 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 |

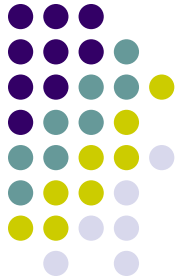
# 簡單的SQL查詢

- 每個簡單查詢範例是針對一個關聯
- Q0：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'
```

  - 查詢的結果可能會有重複的值組

# 簡單的SQL查詢



- Q1：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
```

  - (DNAME=‘Research’)是選擇條件
  - (DNUMBER=DNO)則是合併條件

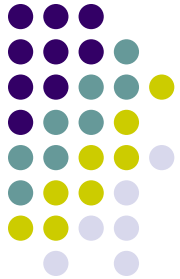
# 簡單的SQL查詢 (續)

- Q2：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

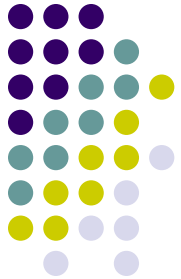
# 修飾屬性名稱



- 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

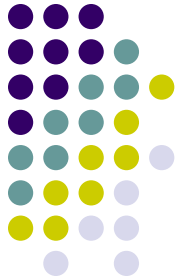


- 有些查詢需要對同一個關聯參考兩次
  - 在這類情況要對關聯名稱指定別名
- Q8：擷取員工的姓名與其直屬上司的姓名

```
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
  - 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是代表扮演上司角色的員工

# 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
- Q9：在資料庫中選擇所有員工的SSN值
  - Q9: SELECT SSN
    FROM EMPLOYEE

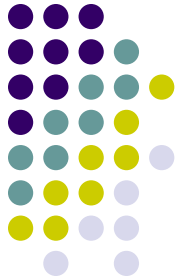
# * 的用法

- 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 (*) 即可代表所有的屬性
  範例：

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

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

# DISTINCT的用法



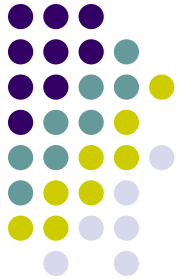
- SQL通常不會將關聯視為集合，因此可以出現重複的值組
- 為了消除查詢結果中的重複值組，可使用關鍵字 **DISTINCT**
- Q11:擷取每一位員工的薪資，以及不同的薪資值。
  - Q11的結果可能會有重複的SALARY值
  - Q11A則不會有任何重複值

```
Q11:    SELECT      SALARY
        FROM        EMPLOYEE
Q11A:   SELECT      DISTINCT SALARY
        FROM        EMPLOYEE
```

# 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

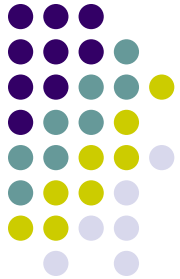
# 子字串比對



- Q12：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‘

```
Q25:    SELECT    FNAME, LNAME
        FROM      EMPLOYEE
        WHERE     ADDRESS LIKE
                  '%Houston,TX%'
```

# 子字串比對



- Q12A：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 (**'+'**、**'-'**、**'*'** 和 **'/'** )
- Q13：顯示出所有工作於 'ProductX' 計畫的員工加薪 10% 後的薪資結果

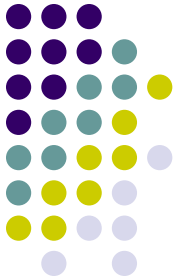
```
Q13:    SELECT    FNAME, LNAME, 1.1*SALARY
        FROM      EMPLOYEE, WORKS_ON,
                  PROJECT
        WHERE     SSN=ESSN AND PNO=PNUMBER
                  AND PNAME='ProductX'
```

# 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
- Q15：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



- 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

# 巢狀查詢

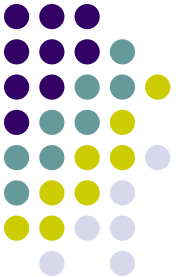
- 所謂的巢狀查詢 (nested query) 是指在一個查詢的WHERE子句內，含有完整的「SELECT-FROM-WHERE」區塊。此時這個外部的WHERE查詢被稱為外部查詢 (outer query)
- Q1：擷取所有在 'Research' 部門工作的員工的姓名與住址

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

# 巢狀查詢

- 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

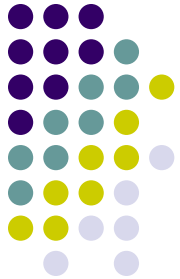
# 相互關聯的巢狀查詢



- 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
- Q16：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.SEX=SEX AND
                    E.FNAME=DEPENDENT_NAME)
```

# 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：

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

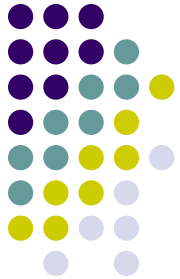
# EXISTS函數

- Q6：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

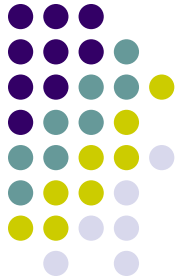
# 明確指定的集合



- 也可以在WHERE子句中，使用明確數值的集合來取代巢狀查詢
- Q17：擷取所有在1、2或3號計畫工作的員工的社會安全號碼

```
Q17:    SELECT    DISTINCT ESSN
        FROM      WORKS_ON
        WHERE     PNO IN  (1, 2, 3)
```

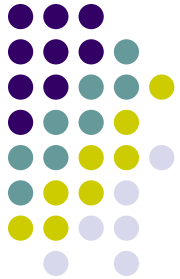
# 聚合函數



- 包括**COUNT**、**SUM**、**MAX**、**MIN**和**AVG**
- Q19：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)
        FROM      EMPLOYEE
```

# 聚合函數



- Q20：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 SUM(SALARY), MIN(SALARY),
                MAX(SALARY), AVG(SALARY)
     FROM EMPLOYEE, DEPARTMENT
     WHERE DNO=DNUMBER  AND
                DNAME='Research'
```

# 聚合函數

- Q21:計算公司裡員工的總數

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

- Q22:在 ‘Research’ 部門工作的員工總數

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

# 分群

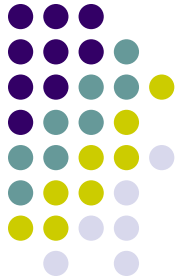
- 在很多時候會需要將聚合函數應用在關聯中，根據某些屬性值分類的值組子群組上
- 每個子群組是由指定的群組化屬性 (grouping attribute) 分組而成，每一組的群組化屬性值是相同的
- 針對每個子群組分別使用函數
- SQL提供**GROUP BY**子句用來指定群組化屬性，它一定要出現在SELECT子句中

# 分組

- Q24：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



- Q24查詢結果

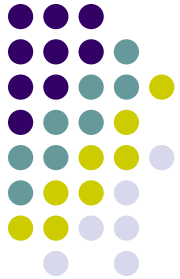
| Dno | Count(*) | Avg(Salary) |
|---|---|---|
| 5 | 4 | 33250 |
| 4 | 3 | 31000 |
| 1 | 1 | 55000 |

# 分組

- Q25：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
```

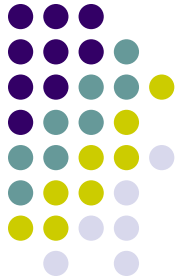
  - 此例的分組與函數動作要等到兩個關聯合併之後才會執行



- Q25查詢結果

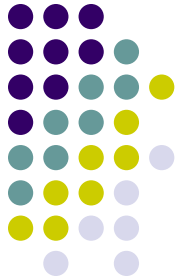
| Pnumber | Pname | Count(*) |
|---|---|---|
| 1 | ProductX | 2 |
| 2 | ProductY | 3 |
| 3 | ProductZ | 2 |
| 10 | Computerization | 3 |
| 20 | Reorganization | 3 |
| 30 | Newbenefits | 3 |

# HAVING子句



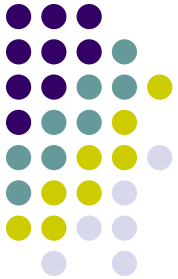
- 有時只需要擷取滿足特定條件群組的函數值
- 此時可使用**HAVING**子句在群組 (而不是個別值組) 上指定選擇條件

# HAVING子句



- Q26：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
```



- Q26查詢結果

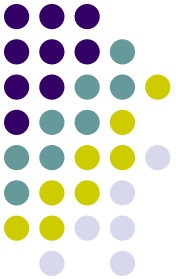
| Pname | Count(*) |
|---|---|
| ProductY | 3 |
| Computerization | 3 |
| Reorganization | 3 |
| Newbenefits | 3 |

# 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：

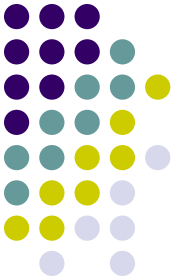
**SELECT** <attribute list>
**FROM** <table list>
[**WHERE** <condition>]
[**GROUP BY** <grouping attribute(s)>]
[**HAVING** <group condition>]
[**ORDER BY** <attribute list>]

# SQL查詢語法總整理



- 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

# 學習評量 (習題)



- 請以SQL語言來撰寫下列針對圖1.2資料庫綱要的查詢。
  a. 擷取所有主修 'CS'(電腦科學) 的大四學生姓名。
  b. 擷取所有在2004及2005年由King教授所開的課程名稱。
  c. 對每一學期由King教授所開的課程，擷取其課程編號、學期 (semester) 、學年 (year) 及修課的學生人數。
  d. 擷取每個主修CS的大四學生 (Class = 4) 的姓名與成績單。成績單內容包括每門已修完的課程名稱、課程編號、學分數、學期、學年及成績。
  e. 擷取每科成績全都為A的所有學生的姓名與主修系所名稱。
  f. 擷取每科成績都不是A的所有學生的姓名與主修系所名稱。

# 學生與課程資訊的資料庫

**STUDENT**

| Name | Student_number | Class | Major |
|---|---|---|---|
| Smith | 17 | 1 | CS |
| Brown | 8 | 2 | CS |

**COURSE**

| Course_name | Course_number | Credit_hours | Department |
|---|---|---|---|
| Intro to Computer Science | CS1310 | 4 | CS |
| Data Structures | CS3320 | 4 | CS |
| Discrete Mathematics | MATH2410 | 3 | MATH |
| Database | CS3380 | 3 | CS |

**SECTION**

| Section_identifier | Course_number | Semester | Year | Instructor |
|---|---|---|---|---|
| 85 | MATH2410 | Fall | 04 | King |
| 92 | CS1310 | Fall | 04 | Anderson |
| 102 | CS3320 | Spring | 05 | Knuth |
| 112 | MATH2410 | Fall | 05 | Chang |
| 119 | CS1310 | Fall | 05 | Anderson |
| 135 | CS3380 | Fall | 05 | Stone |

**GRADE_REPORT**

| Student_number | Section_identifier | Grade |
|---|---|---|
| 17 | 112 | B |
| 17 | 119 | C |
| 8 | 85 | A |
| 8 | 92 | A |
| 8 | 102 | B |
| 8 | 135 | A |

**PREREQUISITE**

| Course_number | Prerequisite_number |
|---|---|
| CS3380 | CS3320 |
| CS3380 | MATH2410 |
| CS3320 | CS1310 |