資料庫系統筆記 Ch. 6

> 參考書籍：
>
> 1. 《Databases System 7th Edition》— Ramez Elmasri, Shamkant B. Navathe.
> 2. 《數據庫系統基礎 第六版》— Ramez Elmasri, Shamkant B. Navathe，李翔鷹、劉鑌、邱海艷、陳立軍譯
>
> 筆記作者：黃漢軒

SQL 資料定義與資料類型

- SQL 分別使用表（Table）、列（Row）和欄（Column），
  來形式化關聯資料模型中的關係（Relation）、元組（Tuple）和屬性（Attribute）
- SQL 的指令結尾需要有一個逗號，代表指令的結束。

SQL 中的模式與目錄的概念

使用 `CREATE SCHEMA 模式名`，可在後面加上 `AUTHORIZATION 使用者名稱`，來宣稱這個模式由哪位使用者來持有。

例如 `CREATE SCHEMA COMPANY AUTHORIZATION 'Jsmith';` 這個語句即為創造一個名為 `COMPANY` 的模式。

SQL CREATE TABLE

語法為 `CREATE TABLE 模式名.關係名`，可以顯式的創立一個在指定模式底下的表，如下

```
CREATE TABLE SCHOOL.STUDENT (
    SID INT NOT NULL,
    Name TEXT NOT NULL,
    PRIMARY KEY (SID)
);
```

若沒有在指定模式底下，則該表稱為基表（Base Table）或基本關聯，如下

```
CREATE TABLE STUDENT (
    SID INT NOT NULL,
    Name TEXT NOT NULL,
    PRIMARY KEY (SID)
);
```

需要注意，創立表時可能會造成錯誤的外來鍵問題，因為另一個表還沒創出來，這時候應該要先把所有的表都創好，再用 `ALTER TABLE` 來將外來鍵加上去。

SQL 的資料類型與值域

值域基本上可以分成：

- 數值型（integer）：包含各式各樣的數值。
- 字串型（string）：可以訂一個固定的長度（使用 `CHAR(n)`），或者訂一個最長的長度（使用 `VARCHAR(n)`），用於儲存字串
- 位元串型（bit-string）：可以訂一個固定的長度（使用 `BIT(n)`），或者訂一個最長的長度（使用 `BIT VARYING(n)`），用於儲存二進制值
- 布林型（boolean）：用於儲存 `TRUE` 或者 `FALSE`
- 日期型（date）：用於儲存日期，有 10 位。
- 時間型（time）：用於儲存時間，有 8 位。
- 時間戳（timestamp）。
- 間隔（interval）。

同時你也可以創造自己的值域，例如學號可能有 10 號，你可以用以下的語句來創立 StudentID 這個值域。

```
CREATE DOMAIN STUDENT_ID AS CHAR(10)
```

在 SQL 中定義約束

指定屬性約束和屬性默認值

- 預設為 `NULL`，但你可以使用 `DEFAULT<VALUE>` 來完成指定默認值。

```
CREATE TABLE EMPLOYEE (
    Dno INT NOT NULL DEFAULT 1
);
```

- 屬性與值域約束可以使用 `CHECK()` 子句來完成。

```
CREATE DOMAIN D_NUM AS INTEGER CHECK(D_NUM > 0 AND D_NUM < 21);
```

指定鍵約束與參照完整性約束

- 可以設定某個屬性為 `PRIMARY KEY`

```
StudentID INT PRIMARY KEY;
```

- 可以設定某個屬性為 `UNIQUE`，限制其屬性唯一性。

```
DName VARCHAR(15) UNIQUE;
```

- 參照完整性主要使用 `FOREIGN KEY` 子句來完成。
  - 對於違反參考完整性的默認動作是拒絕操作，但你可以使用參考觸發動作來指定違反參考完整性的操作，可選 `SET NULL`，`CASCADE` 與 `SET DEFAULT` 這三個操作，選項使用 `ON DELETE` 或 `ON UPDATE` 來限定。
  - 例如 `SET NULL ON DELETE` 與 `CASCADE ON UPDATE` 這兩個操作，第一個操作當東西刪除時會直接將所有的值設成 `NULL`，第二個操作當外碼東西被更新時，會同時更新與其參考的主碼值。
- 約束可以被命名。

元組約束

可在表的最後使用 `CHECK()` 語句來對元組進行約束，

例如 `CHECK(Dept_create_date <= Mgr_start_date)` 就能確保管理員入職的日期必定大於辦公室創立的日期。

基本的 SQL 查詢

- 基本形式：`SELETE <屬性列表> FROM <關係列表> [WHERE <條件表達式>]`

基本的查詢

- 示範一個基本的查詢：

  查詢 John B. Smith 的生日與住址

```
SELECT Bdate, Address
FROM EMPLOYEE
WHERE Fname='John' AND Minit='B' AND Lname='Smith';
```

- 除此之外，你也可以利用關聯的方式來查詢：

```
SELECT Fname, Lname, Address
FROM EMPLOYEE, DEPARTMENT
WHERE Dname='Research' AND Dnumber=Dno;
```

  這方式將 `DEPARTMENT` 表中的 `Dname` 為 `Research`，與 `DEPARTKMENT` 的 `Dnumber` 中等於 `Employee` 的 `Dno` 的資料全部呈現出來。

- 除此之外也可以用限定的方式來避免混淆。

```
SELECT Fname, EMPLOYEE.name, Address
FROM EMPLOYEE, DEPARTMENT
WHERE DEPARTMENT.Dname='Research' AND DEPARTMENT.Dnumber=EMPLOYEE.Dnumber;
```

  這方式可以避免 `DEPARTMENT` 的 `Dnumber` 與 `EMPLOYEE` 的 `Dnumber` 混淆。

- 查詢同一個關係兩次，容易會因為屬性名的多義性而混淆，可以使用 `AS` 來區分。

```
SELECT E.Fname, E.Lname, S.Fname, S.Lname
FROM EMPLOYEE AS E, EMPLOYEE AS S
WHERE E.Super_ssn = S.Ssn;
```

同時可以使用 `EMPLOYEE(Fn, Mi, Ln...)` 來重新命名。

- 選取所有的值，使用星號

```
SELECT * FROM EMPLOYEE WHERE EMPLOYEE.Dno=5;
```

- 使用 `DISTINCT` 來消除重複的元組

```
SELECT DISTIENT Salary FROM EMPLOYEE;
```

這樣就能選取所有薪資的種類。

字串匹配與運算子

- 使用 `%` 來匹配多個字元，使用 `_` 來匹配單個字元

```
SELECT * FROM ACCOUNT WHERE Name LIKE 'Ur_ah';
```

這樣應該會找到 `Uriah` 的資料。

```
SELECT * FROM ACCOUNT WHERE City LIKE 'Tai%';
```

這樣應該會找到 `Tai` 開頭的所有城市名稱。

- 我們可以使用 `ORDER BY` 來進行排序。

```
SELECT * FROM ACCOUNT
ORDER BY Age;
```

這樣應該會提供按照年紀排序的結果。

SQL 的插入、刪除與更新語句

更新

基本形式：`INSERT INTO <屬性> VALUES (...)`，例如：

```
INSERT INTO ACCOUNT
VALUES ("t109590031@ntut.org.tw", "some_hashed_password", 19, "Taiwan", "Taipei");
```

同時需要考慮約束。

刪除

基本形式：`DELETE FROM <屬性> [WHERE <條件判斷式>]`，例如：

```
DELETE FROM ACCOUNT WHERE City='Taipei';
```

即會把所有居住在臺北的帳號刪除，同時請考慮約束。

若不加上條件判斷式，則會清空整個資料表。

更新

基本形式：`UPDATE <屬性> SET <屬性與新值> [WHERE <條件判斷式>]`，例如：

```
UPDATE ACCOUNT SET Age=20, City='Tainan' WHERE Name='Uriah';
```

即會把名子為 `Uriah` 的帳號年紀改為 20 且城市改為臺南。

你可以指定新值為 `NULL` 或 `DEFAULT`，但一句 `UPDATE` 只能更改一個屬性。