Database Management System
Lab 1: Basic SQL

逢甲資工 許懷中

何為 SQL

- 結構化查詢語言(SQL, Structural Query Language)
- IBM 最早採用，原本稱做 IBM Sequel Language
- 1986, ANSI 訂定 SQL 規範，並以之作為關聯式資料庫管理系統的標準語言
  - SQL-86, SQL-89, **SQL-92**, SQL-1999, SQL-2003

用來操作資料庫的語言

- 資料定義語言 (DDL)
  - Data Definition Language
  - 產生資料表、定義 schema
- 資料操作語言 (DML)
  - Data Manipulation Language
  - 查找、結合
- SQL 雖然名字中有 “Query”，但實際上同時包含了 DDL 與 DML

練習環境

- W3C 的 SQL Tryit Editor
- https://www.w3schools.com/sql/
- http://www.w3schools.com/sql/trysql.asp?filename=trysql_select_all
- https://goo.gl/Fg2pod

SQL Basic

QUERY

SELECT 語法

- SELECT *col_name*[, *col_name*] FROM *table_name*;
- SELECT * FROM *table_name*;

Lab 1

- 請練習，由 Customers table 取出 CustomerName 與 City 欄位

- SELECT CustomerName,City FROM Customers;

列出獨一無二的值

- SELECT DISTINCT *col_name*[,*col_name*] FROM *table_name*;

Lab 2

- 請列出顧客們所居住的城市清單(不重複)

- SELECT DISTINCT City FROM Customers;

設定搜尋的條件

- SELECT *col_name*[,*column_name*]
  FROM *table_name*
  WHERE *col_name operator value*;
- Operator =, <>, >, <, >=, <=, BETWEEN, LIKE, IN

試試下列指令

- SELECT * FROM *Customers* WHERE *Country* = 'Mexico';
  - 如果把 = 換成 <> 會如何？
- SELECT * FROM *Customers* WHERE *Country* IN ('Spain', 'Italy');

試試下列指令 (cont.)

- SELECT * FROM Orders WHERE OrderID BETWEEN 10260 AND 10270;
- SELECT * FROM Orders WHERE OrderDate BETWEEN '1996-07-29' AND '1996-08-08';

試試下列指令

- SELECT * FROM Customers
  WHERE City LIKE 's%';
- SELECT * FROM Customers
  WHERE City LIKE '%s';
- SELECT * FROM Customers
  WHERE Country LIKE '%land%';

Lab 3

- 找出在五零年代出生雇員們的名字
- 列出在美國的顧客，其聯絡人的名字與聯絡地址
- 列出單價在 100 元以下的商品
- 列出單價在 10-50 元之間的商品
- 列出以箱 (box) 為單位的商品

Lab 3 (cont.)

- SELECT LastName, FirstName FROM Employees WHERE BirthDate BETWEEN '1950-01-01' AND '1959-12-31';
- SELECT ContactName,Address FROM Customers WHERE Country = 'USA';
- SELECT * FROM Products WHERE price < 100;
- SELECT * FROM Products WHERE price BETWEEN 10 AND 50;
- SELECT * FROM Products WHERE unit LIKE '%box%';

用 AND/OR 連結多個條件

- SELECT * FROM *Products* WHERE *SupplierID* = 20 **AND** Price < 20;
- SELECT * FROM *Customers* WHERE *Country* = 'Germany' **OR** City = 'London';

Lab 4

- 找出由 Employee 4 處理而且由 Shipper 3 運送的訂單
- 找出單價小於 10 或大於 50 的商品

Lab 4 (cont.)

- SELECT * FROM Orders WHERE EmployeeID=4 AND ShipperID=3;
- SELECT * FROM Products WHERE price < 10 or price > 50;
- SELECT * FROM Products WHERE NOT price BETWEEN 10 AND 50;

依序列出

SELECT *col_name*, [*col_name*]
    FROM *table_name*
    ORDER BY *col_name* ASC|DESC[, *col_name* ASC|DESC];

Lab 5

- 依照價格，由低至高列出所有商品的名稱
- 依照年齡，由年輕到老列出所有雇員的姓名

Lab 5 (cont.)

- SELECT ProductName FROM Products ORDER BY Price ASC;
- SELECT LastName,FirstName FROM Employees ORDER BY BirthDate DESC;

集合指令

- SELECT *column_name(s)* FROM *table1*
  UNION/INTERSECT
  SELECT *column_name(s)* FROM *table2*;

試試看下面的指令

- SELECT City FROM Customers
  UNION
  SELECT City FROM Suppliers;
- SELECT City FROM Customers
  INTERSECT
  SELECT City FROM Suppliers;

Lab 6

- 列出德國所有供應商與客戶所在的城市
- 列出德國同時有供應商與客戶所在的城市

Lab 6 (cont.)

- SELECT City FROM Customers WHERE Country = 'Germany'
  UNION
  SELECT City FROM Suppliers WHERE Country = 'Germany';
- SELECT City FROM Customers WHERE Country = 'Germany'
  INTERSECT
  SELECT City FROM Suppliers WHERE Country = 'Germany';

Aggregate

- 彙整一個資料集的值
  - 取平均、最大值、最小值、計數、總數等
- 請嘗試下列 SQL 指令
  - SELECT avg(price) FROM Products WHERE SupplierID=1;
  - SELECT max(price) FROM Products WHERE SupplierID=1;
  - SELECT min(price) FROM Products WHERE SupplierID=1;

Aggregate (cont.)

- 請嘗試下列指令
  - SELECT count(customerID) FROM Orders WHERE EmployeeID=3;
  - SELECT count(distinct customerID) FROM Orders WHERE EmployeeID=3;
  - SELECT count(*) FROM Orders WHERE EmployeeID=3;
  - SELECT sum(Quantity) FROM OrderDetails WHERE ProductID=2;

Lab 7

- 請列出供應商1與3所提供之全部產品的平均價格
- 請列出雇員1至5，其曾經締結訂單的之客戶(不重複)總共有多少位

Lab 7 (cont.)

- SELECT avg(price) FROM Products WHERE SupplierID=1 or SupplierID=3;
- SELECT count(distinct CustomerID) FROM Orders Where EmployeeID BETWEEN 1 and 5;

Aggregate (more...)

- 假如想要對所有供應商、對所有雇員做彙整呢？
- 請嘗試下列指令
  - SELECT SupplierID, avg(price) FROM Products GROUP BY SupplierID;
  - SELECT EmployeeID, count(distinct CustomerID) FROM Orders GROUP BY EmployeeID;

Lab 8

- 請修改前述 SQL 指令，使其同時列出用來 Group 的 Attributes 以及彙整結果
- 試列出各項商品分別的銷售量

Lab 8 (cont.)

- SELECT SupplierID, avg(price) FROM Products GROUP BY SupplierID;
- SELECT EmployeeID, count(distinct CustomerID) FROM Orders GROUP BY EmployeeID;
- SELECT sum(Quantity) From OrderDetails GROUP BY ProductID;

Aggregate (more.)

- 請嘗試下列指令
- SELECT CustomerID, EmployeeID, count(*) as num_trans FROM Orders GROUP BY CustomerID, EmployeeID;

Lab 9

- 試統計透過個別物流業者向不同客戶送貨的次數

Lab 9 (cont.)

- SELECT CustomerID, ShipperID, count(*) as Num_Ship FROM Orders Group By CustomerID, ShipperID;

Aggregate (more...)

- 假如我想對彙整的結果設定條件？
- 請嘗試下列指令
- SELECT CustomerID, EmployeeID, count(*) as num_trans FROM Orders GROUP BY CustomerID, EmployeeID HAVING num_trans > 1;

Lab 10

- 試列出出貨量超過 100 的商品，其商品ID與出貨量
- 試列出達成交易次數不足 10 次的員工

Lab 10 (cont.)

- SELECT ProductID, Sum(Quantity) FROM OrderDetails GROUP BY Product ID HAVING Sum(Quantity) > 100;
- SELECT EmployeeID, count(*) FROM Orders GROUP BY EmployeeID HAVING count(*) < 10;

複合查詢

- 將查詢的結果作為條件的一部分
- 列出所有出貨量大於平均出貨量的訂單明細
- SELECT * FROM OrderDetails WHERE Quantity > (SELECT avg(Quantity) FROM OrderDetails);

複合查詢 (cont.)

- 列出顧客4過去的訂單明細
- SELECT * FROM OrderDetails WHERE OrderID IN (SELECT OrderID FROM Orders Where CustomerID=4);

複合查詢 (cont.)

- 列出顧客4過去的訂單明細
- SELECT * FROM OrderDetails WHERE EXISTS(SELECT * FROM Orders WHERE OrderID=OrderDetails.OrderID AND CustomerID=4);
- SELECT * FROM Orders,OrderDetails WHERE Orders.OrderID=OrderDetails.OrderID AND CustomerID=4;

複合查詢 (cont.)

- 列出僅訂購了一種商品的訂單
- SELECT * FROM Orders WHERE (SELECT count(ProductID) FROM OrderDetails WHERE OrderID=Orders.OrderID GROUP BY OrderID) = 1;
- SELECT * FROM Orders WHERE OrderID IN (SELECT OrderID FROM OrderDetails GROUP BY OrderID HAVING count(productID) = 1);

複合查詢 (cont.)

- 列出出貨總量超過 100 的商品名稱與其銷售量
- SELECT * FROM (SELECT ProductID, Sum(Quantity) as Total_Quantity FROM OrderDetails GROUP BY ProductID) WHERE Total_Quantity > 100;
- SELECT ProductID, Sum(Quantity) FROM OrderDetails GROUP BY ProductID HAVING Sum(Quantity) > 100;

Lab 11

- 列出出貨量最大的出貨明細
- 列出曾經與 顧客4進行交易的員工資料
- 列出委託次數最高的物流廠商ID

Lab 11 (cont.)

- SELECT * FROM OrderDetails WHERE Quantity >= (SELECT max(Quantity) FROM OrderDetails);
- SELECT * FROM Employees WHERE EmployeeID IN (SELECT EmployeeID FROM Orders WHERE CustomerID=4);
- SELECT ShipperID, max(Num_Trans) FROM (SELECT ShipperID,count(*) as Num_Trans FROM Orders GROUP BY ShipperID);

Basic SQL

TABLE UPDATE/DELETE/CREATE

CRUD

插入新的資料列

- INSERT INTO *table_name*
  VALUES (*value1,value2,value3,...*);

- INSERT INTO *table_name* (*column1,column2,column3,...*)
  VALUES (*value1,value2,value3,...*);

Lab 12

- 在 Customer 資料表中加入一列新的顧客資料
  - 顧客名稱: 'Cardinal'
  - 聯絡人: 'Tom B. Erichsen'
  - 地址: 'Skagen 21',
  - 城市: 'Stavanger'
  - 郵遞區號: '4006'
  - 國家: 'Norway';

Lab 12 (cont.)

- INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country) VALUES ('Cardinal','Tom B. Erichsen','Skagen 21','Stavanger','4006','Norway');

變更表格中的資料列

- UPDATE *table_name*
  SET *column1=value1,column2=value2,...*
  WHERE *some_column=some_value*;

Lab 13

- 更新客戶 Alfreds Futterkiste 的資料
  - 將其聯絡人改為 Alfred Schmidt
  - 所在城市改為 Hamburg
- 將所有的商品售價，都調高 30%

Lab 13 (cont.)

- UPDATE Customers
  SET ContactName='Alfred Schmidt', City='Hamburg'
  WHERE CustomerName='Alfreds Futterkiste';

- UPDATE Products SET price=price*1.3

刪除資料列

- DELETE FROM *table_name*
  WHERE *some_column=some_value*;
- 刪去整張資料表內的資料
  - DELETE FROM *table_name*;
  - DELETE * FROM *table_name*;
- 刪去整張 table
  - DROP table *table_name;*

Lab 14

- 將客戶名稱為 Alfreds Futterkiste 同時其聯絡人名稱為 Maria Anders 的資料刪去
- 刪去客戶資料表

Lab 14 (cont.)

- DELETE FROM Customers
  WHERE CustomerName='Alfreds Futterkiste'
  AND ContactName='Maria Anders';

- DROP TABLE Customers

定義一張新的資料表

- CREATE TABLE *table_name*
  (
  column_name1 data_type(size),
  column_name2 data_type(size),
  column_name3 data_type(size),
  ....
  );

試試下列的指令

- CREATE TABLE Persons
  (
  PersonID int,
  LastName varchar(255),
  FirstName varchar(255),
  Address varchar(255),
  City varchar(255)
  );
- INSERT INTO Persons(LastName, FirstName)
  VALUES('John','Smith');

在資料表中加入限制條件

- NOT NULL – 限制某個欄位其值不得為 Null
- UNIQUE – 保證某個欄位中的每一個值皆為唯一
- PRIMARY KEY – 指定某個欄位為主鍵
  - NOT NULL & UNIQUE
- FOREIGN KEY –保證某個欄位的值，必定為參考資料表的主鍵
- CHECK – 確認某個欄位的值要符合特定條件
- DEFAULT – 指定一個欄位的預設值

試試看下列指令

- CREATE TABLE PersonsNotNull (
  P_Id int NOT NULL,
  LastName varchar(255) NOT NULL,
  FirstName varchar(255),
  Address varchar(255),
  City varchar(255));
- INSERT INTO PersonsNotNull(LastName, FirstName) VALUES('John','Smith')

限制欄位值必須獨一無二

- SQL Server/Oracle/MS Access
  - CREATE TABLE Persons (
    P_Id int NOT NULL UNIQUE,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255));

- MySQL
  - CREATE TABLE Persons
    (
    P_Id int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255),
    UNIQUE (P_Id)
    );

- 請試著加資料進入 Persons 資料表

鍵 (Keys)

- 用來分辨一個資料表 (table) 中的不同筆資料

Super Key

- 任何可用來分辨一個資料表中不同筆資料的屬性(們)

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	**Brandt**	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

ID

~~Name~~

Name, dept_name

ID, Name, salary

Candidate Key

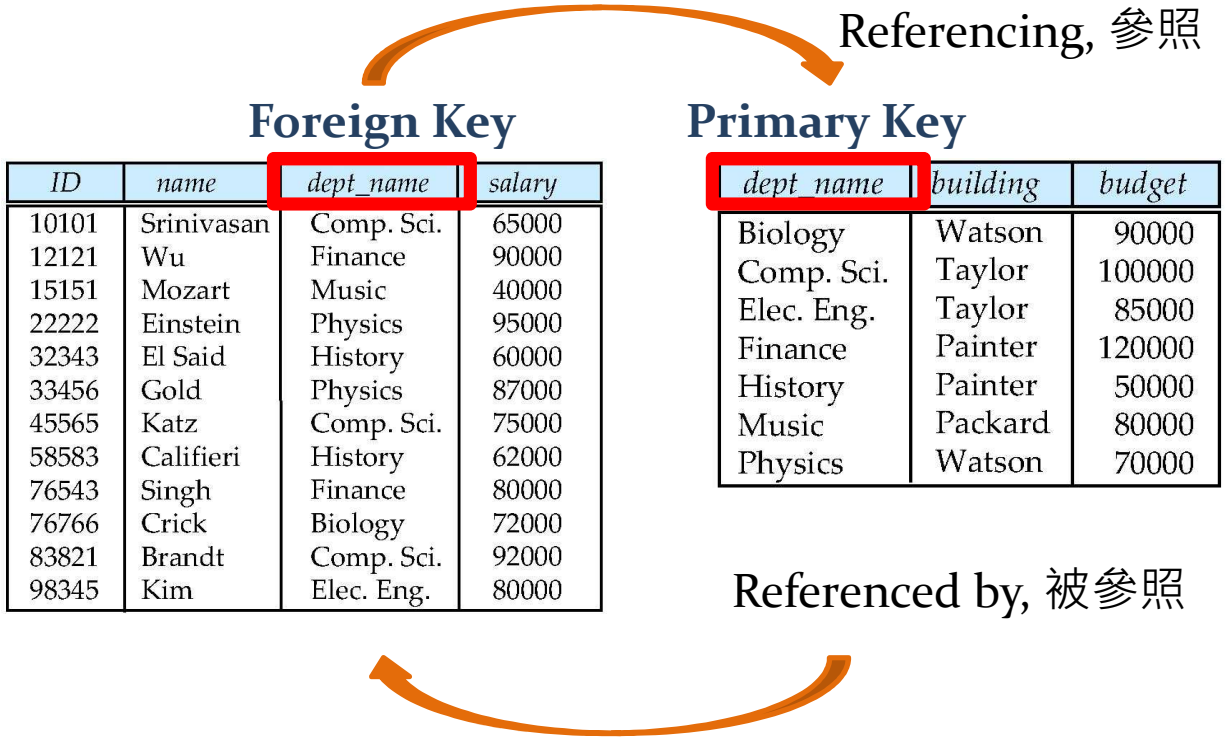
- 成員最少的超鍵 Super Key
  - Candidate Key 必須是 Super Key
  - 候選鍵的子集合不可以是候選鍵
- 對 instructor 資料表 而言
  - Candidate Keys
    - ID
    - name, dept_name
  - ID, name, dept_name 僅僅是個 Super Key
    - 因為 ID 與 name, dept_name 分別都是Candidate Key

Primary Key

- 被選擇來代表一個資料表中各筆資料的 candidate key
- 在一個資料表中，任意資料的 Primary Key 不可重複
  - 這是一個約束 (constraint)
  - 表示任何對於該資料庫的操作，都不可違背
- 避免選擇可能會變更的 Candidate Key 為 Primary Key

外來鍵 (Foreign Key)

- 用來參照其他關聯 (relation) 的鍵 (key)
- 必須是被參照關聯的主鍵 (Primary Key)



ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

dept_name	*building*	*budget*
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

設定主鍵

- SQL Server/Oracle/MS Access
  - CREATE TABLE Persons(
    P_Id int NOT NULL PRIMARY KEY,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255));
- MySQL
  - CREATE TABLE Persons(
    P_Id int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255),
    PRIMARY KEY (P_Id));

設定外來鍵

- SQL Server/Oracle/MS Access
  - CREATE TABLE OrdersPerson(
    O_Id int NOT NULL PRIMARY KEY,
    OrderNo int NOT NULL,
    P_Id int FOREIGN KEY REFERENCES
    Persons(P_Id)) ;
- MySQL
  - CREATE TABLE OrdersPerson(
    O_Id int NOT NULL,
    OrderNo int NOT NULL,
    P_Id int,
    PRIMARY KEY (O_Id),
    FOREIGN KEY (P_Id) REFERENCES
    Persons(P_Id));

Check

- SQL Server/Oracle/MS Access
  - CREATE TABLE Persons(
    P_Id int NOT NULL CHECK (P_Id>0),
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255));
- MySQL
  - CREATE TABLE Persons(
    P_Id int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255),
    CHECK (P_Id>0));

Default

- CREATE TABLE Persons(
  P_Id int NOT NULL,
  LastName varchar(255) NOT NULL,
  FirstName varchar(255),
  Address varchar(255),
  City varchar(255) DEFAULT 'Sandnes')

More Practices

- http://www.w3schools.com/sql/sql_quiz.asp
- http://sqlzoo.net/wiki/SQL_Tutorial/zh