**NoSQL(非關聯式資料庫)**

1. 不需要事先定義欄位,因此就算不同資料row間欄位有差異,都還是可以整合在同一個表格中,對於非結構或半結構式資料友善

* **結構化資料 : Oracle、PostgreSQL**
* **非結構化資料 : Redis、MongoDB**

1. **MongoDB**

預設PK : **\_id**

Table : 預設叫**Collection**

資料 : 預設叫**Document**

**Robo (連線至MongoDB,並輸入語法以取用的媒介)**

1. 左上角”connect” : 可建立與本地端的連線
2. **創建資料庫** : “Connection名”按右鍵 — “Create Database”
3. **創建表格** : “Database名”按右鍵 — “Create Collection”
4. **語法**

* show dbs : 查全部有哪些資料庫
* show collections : 查全部有哪些表格
* use 資料庫名 : 創建新資料庫
* db : 顯示現在的表格是在哪個資料庫
* db.表格名(沒有就創建).insertMany(資料dict) : 插入(多筆)資料
* db.表格名.count() : 看該表格有幾個row
* db.表格名.drop() : 刪除表格
* db.dropDatabase() : 刪除整個資料庫
* **查找資料**

1. 常用符號

* 比較
* $gt : 大於
* $lt : 小於
* $gte : 大於等於
* $eq : 等於
* $ne : 不等於
* $in : 包含
* 邏輯
* $and
* $not
* $nor
* $or
* $all
* 其他
* $exists : 存不存在
* $elemMatch : 看陣列裡有沒有匹配的值
* $type : 指定型別
* $size : 指定要找內容有幾筆的資料列
* $regex: : 查正規表達式

1. 用法

* db. getCollection('表格名').find({"xxxxx.xx":"xxxxxx"})

= db.表格名.find({"xxxxx.xx":"xxxxxx"})

* db.表格名.find({欄位名 : {$eq : 值}})

或 db.表格名.find({欄位名 :值})

* db.表格名.find({欄位名 : {$nin : ['值','值']}})
* db.表格名.find({$and : [{條件1},{條件2}]})

或 db.表格名.find({條件1,條件2})

* db.表格名.find({欄位名 : {$not:{$gt : 值}}})
* db.表格名.find({欄位名 : {$exists : false}})
* db.表格名.find({欄位名 : {$elemMatch : {陣列的Key : Value}}})
* db.表格名.find({欄位名 : {$size : 2}})
* db.表格名.find({欄位名 : {$type : "int"}})
* db.表格名.find({"欄位名":{$regex: /要找的文字/ i }})

= db.表格名.find({"欄位名":/要找的文字/ i })

>> 選擇性填寫option包含 : i (大小寫視為一樣)、x (忽略空字元)

1. 其他

* 指定只顯示某欄位 :

db.表格名.find({…},{“欄位名” :1 ,”\_id” : 0}) >>(顯示為1)

* 顯示筆數 : db.表格名.find({…}).limit(筆數)
* 忽略前面幾筆 : db.表格名.find({…}).skip(筆數)
* 排序 : db.表格名.find({…}).sort({欄位名:1}) >>(由大到小是-1)
* **更新資料**
  + - 指定更改欄位

db.表格名.update({欄位名.column名:" xxxxxx "},{$set:{欄位名.$.column1名:"xxx", 欄位名.$.column2名:" xxxxxx "}}},{multi: true , upsert: false})

>>> $表示同一層

>>> multi : 預設false，只更新找到的第一筆紀錄

>>> upsert : 如果不存在則新增

* + - 不指定更改欄位

db.表格名.update({},{$set:{欄位名:{ column1名:"xxx"}}})

>>> 會改到多筆的話要用updateMany(相當於multi: true)

* + - db.表格名.save({更改或新增資料})
    - 某欄位要多新增一個值進陣列

db.表格名.update({…},{$push:{….}})

* + - 某欄位要多新增一個陣列的所有值

db.表格名.update({…},{$push:{欄位名:{$each:[陣列]}}})

* + - 每次修改都累加

db.表格名.update({…},{"$inc" : {欄位名 : 1}})

>>> 每次執行結果 : 2 > > 3 > > 4

* + - 將欄位的值設定為當前日期

db.表格名.update({…},{$currentDate :{欄位名:{$type:"date"}}})

* + - 將低於門檻值的欄位，拉高至門檻值

db.表格名.update({…},{$max :{欄位名: "門檻值"}}})

* + - 將欄位的值乘以某倍數

db.表格名.update({…},{$mul :{欄位名:”倍數”}}})

* + - 重新命名欄位

db.表格名.update({…},{$rename :{欄位名:”新名”}}})

* + - 刪除欄位

db.表格名.update({…},{$unset :{欄位名:””}}})

* **移除資料** : db.表格名.remove({"xxxxx.xx":"xxxxxx"})
* **Aggregate(篩選)**

1. 有點像查詢,但是可以一次用多個篩選條件
2. db.表格名.aggregate([{篩選條件一},{篩選條件二},{篩選條件三}])
3. 範例

* 選擇要顯示的欄位

db.表格名.aggregate([{**$project** : {欄位名: 1}}])

* 篩出匹配條件的

db.表格名.aggregate([{**$match** :{條件}}])

* 把document依指定欄位分組呈現,並計算某欄值的總和

db.表格名.aggregate([{**$group** :{"新的欄位名1(自己取)" : "$要分組的欄位名1","新的欄位名2":{$sum : "$要分組的欄位名2"}}}])

* 把為陣列的值展開成各一個document

db.表格名.aggregate([{**$unwind** : "**$**欄位名"}])

* 排序(由小到大)

db.表格名.aggregate([{**$sort**: {欄位名: 1}}])

* 只顯示前2筆

db.表格名.aggregate([{**$limit**: 2}])

* 忽略前3筆

db.表格名.aggregate([{**$skip**: 3}])