**1. 簡介關聯式/非關聯式資料庫(NoSQL、NonSQL)**

**SQL (關聯式) 與NoSQL (非關聯式) 資料庫的比較**

**幾十年來，用於開發應用程式的主要資料模型，均是諸如 Oracle、DB2、SQL Server、MySQL 和 PostgreSQL 等關聯式資料庫所採用的關聯式資料模型。一直到 2000 年中後期，其他資料模型才開始廣受採納和運用。為了區分和歸類這些新類型的資料庫和資料模組，因此創造了「NoSQL」這個名詞。「NoSQL」這個名詞常常與「非關聯式」互換使用。**

**雖然 NoSQL 資料類型多樣且功能各異，但您可從以下表格瞭解 SQL 和 NoSQL 資料庫的一些差異性。**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | 關聯式資料庫 | NoSQL 資料庫 | | 最佳工作負載 | **關聯式資料庫專門用於交易性以及高度一致性的線上交易處理 (OLTP) 應用程式，並且非常適合於線上分析處理 (OLAP) 使用。** | **NoSQL 資料庫專門用於包含低延遲應用程式的多樣資料存取模式。NoSQL 搜尋資料庫專門用於進行半結構資料的分析。** | | 資料模型 | **關聯式模型將資料標準化，成為由列和欄組成的表格。結構描述嚴格定義表格、列、欄、索引、表格之間的關係，以及其他資料庫元素。此類資料庫強化資料庫表格間的參考完整性。** | **NoSQL 資料庫提供鍵值、文件和圖形等多種資料模型，具有最佳化的效能與規模。** | | 效 能 | **一般而言，效能取決於磁碟子系統。若要達到頂級效能，通常必須針對查詢、索引及表格結構進行優化。** | **效能通常會受到基礎硬體叢集大小、網路延遲，以及呼叫應用程式的影響。** | | 擴 展 | **關聯式資料庫通常透過增加硬體運算能力向上擴展，或以新增唯讀工作負載複本的方式向外擴展。** | **NoSQL 資料庫通常可分割，因為存取模式可透過使用分散式架構來向外擴展，以近乎無限規模的方式提供一致效能來增加資料吞吐量。** | | API | **存放和擷取資料的請求是透過符合結構式查詢語言 (SQL) 的查詢進行通訊。這些查詢是由關聯式資料庫剖析和執行。** | **以物件為基礎的 API 讓應用程式開發人員可輕鬆存放和擷取資料結構。應用程式可透過分區索引鍵查詢鍵值組、欄集，或包含序列化應用程式物件與屬性的半結構化文件。** | |  |

**資料來源:** [**https://aws.amazon.com/tw/?nc2=h\_lg**](https://aws.amazon.com/tw/?nc2=h_lg)

**2. 詳細介紹任一非關聯式資料庫**

**MongoDB：開源、好上手的 NoSQL 資料庫**

****

**MongoDB 是個開源的 NoSQL 資料庫，由於你不需要花很多時間先制定每張資料表要長怎樣、畫出 ER Model，因此不管是用來做 Side-project，還是用來塞大量資料都很適合。**

**MongoDB 是個開源的 NoSQL 資料庫，由於你不需要花很多時間先制定每張資料表要長怎樣、畫出 ER Model，因此不管是用來做 Side-project，還是用來塞大量資料都很適合。**

**MongoDB 是什麼**

**NoSQL 的全名是 Not only SQL。而 MongoDB 的 Logo 是個葉子，並不是一顆芒果，仔細看它是 Mongo 而不是 Mango，這邊不要搞錯了。**

***就像 Kangaroo 的中文並不是長頸鹿。***

**MongoDB 是用 Key-value 的方式來儲存資料，長得大概像這樣子：**

**{**

**name: "NoobTW",**

**url: "https://noob.tw"**

**}**

**長的就跟 JSON 沒什麼兩樣。值得注意的是：MongoDB 用的是所謂 BSON 而不是 JSON、每筆資料的 key 和 value 都是區分大小寫的。**

**BSON 是 Binary JSON *的縮寫*，就是拿 JSON *下去擴充*，*所以可以塞* Binary data *等* JSON *不能塞的東西*。**

**在 MongoDB 中，每筆資料稱為一個 Document，大概就是對應到傳統資料庫的 Row。而裝這些 Documents 的叫做 Collection，也差不多可以對應到傳統資料庫的 Table。**

**資料庫則是一樣叫 Database 了。**

**安裝 MongoDB**

**一樣先用 apt-get 安裝 MongoDB。**

sudo apt-get install mongodb

**操作 MongoDB Shell**

**在用 JS 操作前，我們先學會用 Mongo Shell 操作看看吧。輸入 mongo 進到 Mongo 的 Shell。**

**想要查看所有資料庫，可以使用這個指令：**

show dbs;

**一開始應該只會有兩個預設的資料庫：**

admin 0.000GB

local 0.000GB

**如果我們要使用一個新的資料庫的話，例如我們今天假設要建立一個部落格系統，要開一個新資料庫叫 blog，那可以直接使用這個指令：**

use blog;

**不需要管這個資料庫有沒有存在，只要不存在的話它會自動幫你建立。接著我們來試著新增一筆資料，假設我們要新增一個文章資料的話，那一樣我們需要一個 Collection 叫 articles，不需要特別建立，可以直接使用 insert 來新增一筆資料（document）：**

# db.collection.insert(新增內容);

db.articles.insert({

title: '測試文章',

body: '測試內容',

time: new Date('2019-09-29 00:00:00'),

author: 'NoobTW',

});

**會回傳一個：**

WriteResult({ "nInserted" : 1 });

**代表我們成功寫入了一筆資料。**

**我們可以使用 find 來查資料：**

# db.collection.find(查詢條件)

db.articles.find({});

**中間的 {} 是不設查詢條件的意思，應該會看到一筆我們剛剛新增的資料：**

{ "\_id" : ObjectId("5d90a6d70002d2373bcb13b0"), "title" : "測試文章", "body" : "測試內容", "time" : ISODate("1970-01-01T00:00:00Z"), "author" : "NoobTW" }

**其中 \_id 是 MongoDB 自動產生的 ID，每一筆資料（document）都會產生一個獨一無二的 ID。**

**如果要更新資料的話，例如我們要把標題「測試文章」這筆資料改成標題「測試標題」，那可以這樣下：**

# db.collection.update(查詢條件, 新資料)

db.articles.update({title: '測試文章'}, {$set: {title: '測試標題'}});

**前面是查詢條件（你在 find 也可以這樣下），後面是新資料內容，其中 $set 是只修改指定欄位的意思。如果你沒有加 $set 的話，這筆資料會只剩下 title 這個欄位，其他欄位都不見了。**

**最後如果是要刪除資料的話：**

# db.collection.remove(查詢條件)

db.articles.remove({title: '測試文章});

**這樣就能把標題為「測試文章」的所有資料都刪除掉。**

**這篇講了簡單的 Mongo Shell 的 CRUD 操作。由於官方文件說：**

**The mongo shell is an interactive JavaScript interface to MongoDB.  
（Mongo shell 是一個存取 MongoDB 的互動 JavaScript 介面。）**

**換句話說，當你熟悉 Mongo Shell 以後，未來在 Node.js 上操作 MongoDB 也會相當熟練，畢竟語法大同小異。可以試著多開幾個不同的 Collection 操作看看，也可以去研究怎麼操作複雜的 Array，如搭配 $push 來 update、或使用 $and 來 find 等等，後面的文章會探討如何備份還原 MongoDB 資料庫，以及如何在 Node.js 上操作。**

**資料來源:** [**https://noob.tw/mongodb/**](https://noob.tw/mongodb/)  [iT邦幫忙](https://ithelp.ithome.com.tw/articles/10222561)

**3. 請比較關聯式/非關聯式資料庫優缺點**

SQL作為主要的資料儲存方式已經超過40年，並且經歷了至少兩個指數擴張期：20世紀90年代Web應用程式崛起之後，以及在過去十年中由於移動裝置爆炸引起的擴張。 因此，越來越小的公司開始發現使用資料庫的好處，而像Google這樣的網際網路巨頭則已經將資料單位上升到PB或甚至EB。

在SQL 的發展過程中，產生了許多迭代產品，其中最重要的是SQL，NoSQL和NewSQL — 它們一起負責絕大部分的資料庫市場。

所以，如果你要選擇一個數據庫工具，你應該選擇哪一個？ 其實，沒有明確的答案。 不同的人和公司選擇不同，這更多地取決於他們對每個特定專案的偏好和相對優勢，而不是其中一個對其他所有的直接優勢。 那麼，這些優點和缺點是什麼？ 讓我們來看一下。

**SQL**

SQL是關係型資料庫管理系統（RDBMS），顧名思義，它是圍繞關係代數和元組關係演算構建的。 70年代以來，它一直是主要的資料庫解決方案，只是最近才有了其他產品的空間。 不管有些人說什麼，這意味著它一直能出色地執行廣泛的任務。 其主要優點如下：

* 不同的角色（開發者，使用者，資料庫管理員）使用相同的語言。
* 不同的RDBMS使用統一標準的語言。
* SQL使用一種高階的非結構化查詢語言。.
* 它堅持 [ACID](http://searchsqlserver.techtarget.com/definition/ACID) 準則 (原子性，一致性，隔離性，永續性),，這些準則保證了資料庫尤其是每個事務的穩定性，安全性和可預測性。

如你所見，許多SQL的好處來源於它的統一性，舒適性和易用性。 即使你只有非常有限的SQL知識（或完全沒有，如果需要），你可以在像 [online SQL Query Builder](https://www.datapine.com/sql-query-builder) 這樣的特殊工具幫助下使用它。

然而，它的缺點使得它非常不適合某些型別的專案。 SQL的主要問題是它難以擴充套件，因為它的效能隨著資料庫的變大而快速下降。 分散式也是有問題的。

NoSQL和NewSQL出現的原因之一是，以前的RDBMS的設計不能滿足現代資料庫每秒處理的事務數量。 像亞馬遜或阿里巴巴等需要處理驚人資料量的巨頭，以前的RDBMS會在幾分鐘內出現問題。

**NoSQL (Not Only SQL)**

NoSQL越來越受歡迎，其中最重要的實現是Apache Cassandra，MongoDB等產品。 它主要用於解決SQL的可擴充套件性問題。 因此，它是沒有架構的並且建立在分散式系統上，這使得它易於擴充套件和分片。

然而，這些好處是以放寬ACID原則為代價的：NoSQL採取最終一致性原則，而不是所有四個引數在每個事務中保持一致。 這意味著如果在特定時間段內沒有特定資料項的更新，則最終對其所有的訪問都將返回最後更新的值。 這就是這樣的系統通常被描述為提供基本保證的原因（基本可用，軟狀態，最終一致性） — 而不是ACID。

雖然這個方案極大地增加了可用時間和伸縮性,它也會導致資料丟失----這個問題的嚴重程度取決於資料庫伺服器的支援情況和應用程式碼質量.在某些情況下,這個問題十分嚴重.

另一個NoSQL出現的問題是現在有很多型別的NoSQL系統,但它們之間卻幾乎沒有一致性.諸如靈活性,效能,複雜性,伸縮性等等特性在不同系統間差別巨大,這使得甚至是專家在他們之間都很難選擇.不過,當你根據專案特點作出了合適的選擇,NoSQL可以在不顯著丟失穩定性的情況下提供一個遠比SQL系統更高效的解決方案.

**NewSQL**

NewSQL是一種相對較新的形式，旨在使用現有的程式語言和以前不可用的技術來結合SQL和NoSQL中最好的部分。 NewSQL目標是將SQL的ACID保證與NoSQL的可擴充套件性和高效能相結合。

顯然，因為結合了過去僅單獨存在的優點，NewSQL看起來很有前途; 或許，在未來的某個時候，它將成為大多數人使用的標準。 不幸的是，目前大多數NewSQL資料庫都是專有軟體或僅適用於特定場景，這顯然限制了新技術的普及和應用。

除此之外，NewSQL在每個方面比較均勻，每個解決方案都有自己的缺點和優勢。 例如，SAP HANA可以輕鬆處理低到中等的事務性工作負載，但不使用本機叢集，MemSQL對於叢集分析很有用，但在ACID事務上表現出較差的一致性，等等。 因此，在這些解決方案變得真正普及之前，可能還需要一段時間。

**結論**

圍繞SQL有許多謬見和誤解：例如，SQL已過時，應該儘可能替換為NoSQL或New SQL。 當然，這不是真的。 目前，在三種基本替代方案中沒有明確的領導者 - 每一種都有更適合的專案，而在其他情況下不太適合（或完全不適合）。 因此，沒有普遍的理想選擇。 例如，如果你主要考慮資料庫應始終可用於接受新的內容，則應考慮最終一致性解決方案，如Cassandra或Riak。 如果你追求快取記憶體SQL，新的快取資料庫比如VoltDB似乎是明智的選擇; 等等。

**資料來源:https://www.itread01.com/content/1542393729.html**