

東海大學資訊工程學系研究所

碩士論文

指導教授：陳隆彬博士

基於 MongoDB 建置電腦圍棋棋譜搜尋系統

**A mongoDB based computer GO searching
system**

研究生：張嘉麟

中華民國一百零四年七月

東海大學碩士學位論文考試審定書

東海大學資訊工程學系 研究所

研究生 張嘉麟 所提之論文

基於 MongoDB 建置電腦圍棋棋譜搜尋系統

經本委員會審查，符合碩士學位論文標準。

學位考試委員會

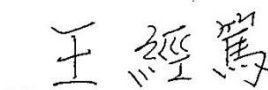
召集人



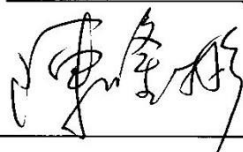
簽章

委員

員



指導教授



簽章

中華民國 104 年 6 月 18 日

摘要

棋譜搜尋功能是電腦奕棋的重要基礎，本論文研究基於 MongoDB 的棋譜搜尋系統。論文首先討論棋譜格式作的前置處理，將網際網路上的文字棋譜資訊擷取並轉換為資料庫的棋盤盤面表示法。本論文開發出一個方法，把一個完整的圍棋盤面切割成所有可能的位移和旋轉的小盤面，再儲存至資料庫中，此方法能讓使用者執行一次搜尋便可以找出所有符合條件的盤面。由於棋譜資料庫的資訊單純，本論文採用 MongoDB 雲端資料庫作為圍棋資料庫，來取代坊間常用的關聯性資料庫的角色，發揮 MongoDB 於巨量資料處理上的優勢，借以提高系統效率。本系統可從棋譜中找出常見棋形，讓圍棋程式能活用於盤面的判斷。

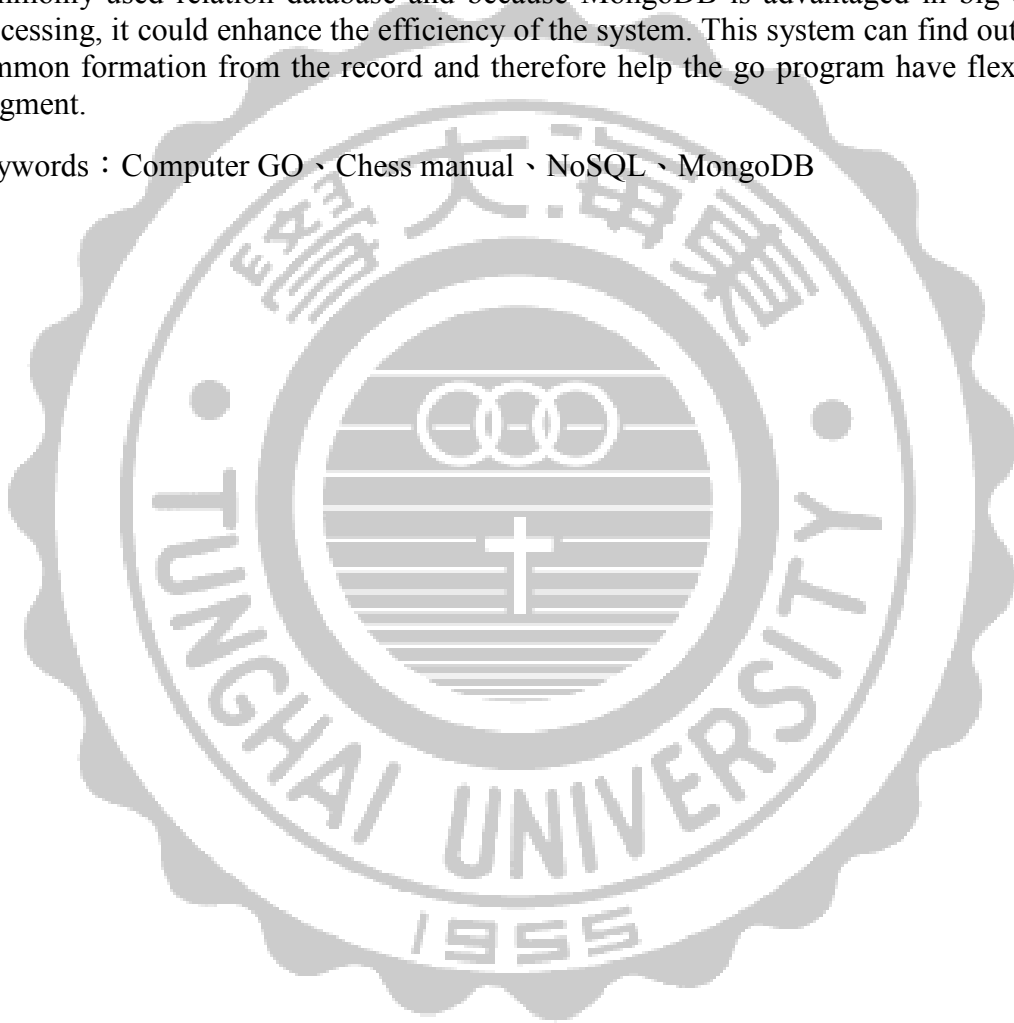
關鍵詞：電腦圍棋、棋譜、NoSQL、MongoDB



ABSTRACT

Go record searching is the important foundation of Computer Go. This study is based on the high-efficient go record searching system of MongoDB. Firstly, this study collected go record manuals on the Internet and transformed them into go board notation for the pre-process of go record formation. This study developed a method that cuts the go board into small pieces that contain every possible shifts and rotations in the go game and then saved them in the database. This method can help the user to find out all the boards that matched the conditions in a search. Owing to the simplicity of go record database, this study adopted the MongoDB cloud database to replace the commonly-used relation database and because MongoDB is advantaged in big data processing, it could enhance the efficiency of the system. This system can find out the common formation from the record and therefore help the go program have flexible judgment.

Keywords : Computer GO 、 Chess manual 、 NoSQL 、 MongoDB



目錄

摘要.....	3
ABSTRACT.....	4
第一章、緒論(Introduction).....	8
第二章、背景(Background).....	10
2.1 棋譜的 SGF 格式.....	10
2.2 MongoDB 資料庫.....	11
第三章、系統架構與設計.....	17
3.1 系統架構.....	17
3.2 SGF 棋譜擷取.....	17
3.3 棋形的轉換切割和導入資料庫.....	18
3.3.1 棋形轉換.....	18
3.3.2 盤面切割.....	19
3.3.3 資料庫欄位.....	21
3.3 棋形搜尋.....	22
第四章、系統實作.....	25
4.1 SGF 檔案分析處理程式.....	25
4.2 導入 MongoDB 之程式.....	27
MongoDB API.....	28
4.3 棋形搜尋程式.....	35
4.4 棋形搜尋驗證.....	36
第五章、結論與展望.....	43
第六章、參考文獻.....	44

圖表目錄

圖表 1 棋譜記錄擷取與查詢系統.....	8
圖表 2 SGF 檔案內容之舉例.....	10
圖表 3 SGF 棋譜內容縮寫例解.....	10
圖表 4 SGF 位置表示範例.....	11
圖表 5 mongoDB 資料模型.....	12
圖表 6 關連式資料庫與 MongoDB 使用術語差異.....	12
圖表 7 MongoDB 結構.....	13
圖表 8 Jason 與 Bson 的差異比較.....	14
圖表 9 sharding 架構圖.....	15
圖表 10 shard 分配 chunk 示意圖.....	16
圖表 11 SGF 檔案分析處理流程圖.....	17
圖表 12 SGF 棋譜賽事與著手資訊.....	18
圖表 13 數字盤面示意圖.....	19
圖表 14 圖表 13 之前五步著手(B[qd];W[dp];B[dd];W[kp];B[pe];)示意圖.....	19
圖表 15 2X2 棋形變化.....	20
圖表 16 圍棋手段的常見棋形範例.....	20
圖表 17 3X3 棋形切割示意圖.....	20
圖表 18 棋形的旋轉角度與鏡射示意圖.....	21
圖表 19 gotable 棋譜資料庫資料型態表.....	22
圖表 20 棋形搜尋流程圖.....	22
圖表 21 相異座標但相同棋形之示意圖.....	23
圖表 22 棋形搜尋的分割與座標分配示意圖.....	23
圖表 23 3X3 分割棋形的搜尋流程圖.....	24
圖表 24 棋譜搜尋系統資料庫架構圖.....	27
圖表 25 mongoDB 初始化.....	27
圖表 26 mongoDB 初始化完成.....	28
圖表 27 MongLab 線上 MongoDB 服務畫面.....	30
圖表 28 Mongolab 建立 database.....	31
圖表 29 Mongolab 管理界面.....	31
圖表 30 Mongolab Collection 管理畫面.....	32
圖表 31 Mongolab 建立 Document 畫面.....	33
圖表 32 Mongolab 查詢棋型畫面.....	34
圖表 33 棋形搜尋程式操作流程.....	35
圖表 34 系統搜尋介面.....	35
圖表 35 MongoLab 新增查詢.....	36

圖表 36	MongoLab 輸入搜尋畫面.....	37
圖表 37	MongoLab 搜尋結果 1.....	38
圖表 38	Lgs. tw 線上棋譜編輯器 1.....	39
圖表 39	MongoLab 搜尋結果 2.....	39
圖表 40	Lgs. tw 線上棋譜編輯器 2.....	40
圖表 41	Lgs. tw 線上棋譜編輯器 3.....	40
圖表 42	MongoLab 搜尋結果 3.....	41
圖表 43	Lgs. tw 線上棋譜編輯器 4.....	41
圖表 44	Lgs. tw 線上棋譜編輯器 5.....	41



第一章、緒論(Introduction)

在人工智慧的領域內，電腦對局有著相當重要的地位。電腦對局中包含了西洋棋、象棋、圍棋等各種棋類遊戲。西洋棋和象棋的電腦對弈程式很早就已經可以達到該領域高段職業棋手的水準。在 1997 年，由 IBM 的深藍程式(Deep Blue)以二勝一敗二和的成績擊敗當時的西洋棋棋王 Garry Kasparov，獲得全世界的關注。相較於西洋棋，電腦圍棋則是近幾年才達到職業水準的程度。

遊戲樹搜尋是電腦奕棋人工智慧程式的基礎。每一個遊戲樹節點代表一個盤面(state)，所有盤面形成一個狀態空間(state space)。一個遊戲樹節點的下一層節點為該盤面的下一步所有可能的走法。節點的分支度(branching factor)是所有可能的走法的數量。為增加棋力，電腦程式必須探索遊戲樹的足夠的深度才能分析多步之後的對局變化。因此，電腦奕棋的複雜度(complexity)受到遊戲樹的深度與分支度的影響。圖表 1 棋譜記錄擷取與查詢系統顯示常見電腦對局的複雜度。

Game	Board size	State-space complexity	Branching factor	Complexity class
Tic-tac-toe	9	3	4	PSPACE-complete
Connect Four	42	13	4	PSPACE
Chinese checkers 中國跳棋 (2 sets)	121	23		EXPTIME-complete
Hex (11x11)	121	57	280	PSPACE-complete
Gomoku 五子棋 (15x15)	225	105	210	PSPACE-complete
Chess(西洋棋)	64	47	35	EXPTIME-complete
Connect6	361	172	46000	PSPACE-complete
Xiangqi (象棋)	90	40	38	believed to be EXPTIME-complete
Shogi	81	71	92	EXPTIME-complete
Go (圍棋) (19x19)	361	171	250	EXPTIME-complete
Bejeweled and Candy Crush (8x8)	64	<50		NP-hard

圖表 1 棋譜記錄擷取與查詢系統

圍棋雖然規則簡單，只要圍的地越大的那一方就代表勝利，但實際玩起來卻不是那麼一回事。由於棋盤廣大，正式對局都是以 19X19 的大小進行，複雜度可說是千變萬化，亦不可能下出一模一樣的走法。人類在下圍棋時是藉由盤面情勢以及經驗來判斷下一步該走哪裡，也就是藉由不斷地對奕來產生所謂的”棋感”。棋感越好的人，棋力就越強，而電腦圍棋也是需要倚賴專家的經驗和知

識來輔助它成長。若是能將古今中外，大量的職業棋士競賽的棋譜建立成資料庫，可以更有系統地從中獲取一些資訊，進而提升程式的棋力。

棋形比對在電腦奕棋是一個相當重要的功能，尤其對於開局和殘局而言具有決定性影響。另外，電腦分析工具亦大量借重棋形比對來找出各種策略之相關性。本論文將建立棋形搜尋系統，讓使用者輸入棋形，來從棋譜資料庫找尋相關的棋譜。

本論文設計並實作一個電腦棋譜搜尋系統。本系統包含了以下三個模組：

- 棋譜擷取模組：此模組將網際網路上的棋譜資訊擷取並轉換為制式表示法。由於網路上棋譜不斷演進，我們發展自動化工具來擷取棋譜。
- 資料轉換與儲存模組：此模組把一個完整的圍棋盤面切割成所有可能的位移和旋轉的小盤面，然後儲存至資料庫。
- 棋譜搜尋模組：此模組讓使用者透過圖形化介面輸入盤面樣式。此模組進行資料庫搜尋找出所有符合條件的盤面，並顯示在圖形化介面中。

電腦最擅長的就是在短時間內執行大量的搜尋，電腦圍棋中更是需要在短時間搜尋處理龐大的資料，從棋譜資料庫中檢出某種特定的棋形、定石來判斷來手的位置。由於棋譜資料具有資訊單純和資料龐大的特性，本文提出了以 NoSQL 型態的 MongoDB 來建設棋譜搜尋系統。NoSQL 資料庫有一個重要的特性是具備水平擴充的能力，只要新增服務節點，就可以不斷擴充資料庫的容量，並且可以利用低價一般等級的電腦就能進行水平擴充，這是傳統關聯式資料庫無法達到的。傳統資料庫的叢集系統需要效能和容量較大的伺服器才能勝任，而 NoSQL 資料庫可以用更低廉的成本打造出 TB 或是 PB 等級的大型資料庫。

本論文採用 MongoDB 雲端資料庫作為圍棋資料庫，來取代坊間常用的關聯性資料庫的角色，我們使用 JAVA 為主要的程式設計出圖形化介面，讓使用者輸入欲尋的棋形，再讓系統從眾多賽事棋譜中找出相符的棋形，最後回傳該棋形的相關賽是資訊給使用者。

本論文的組織章節如下，第二章介紹相關技術，第三章則講述系統設計過程與實作，第四章展現實驗結果與分析，第五章歸納出結論及未來工作。

第二章、背景(Background)

2.1 棋譜的 SGF 格式

本論文所採用的棋譜格式為SGF，即Smart Game Format 的縮寫（以下簡稱為SGF），SGF 是由Auder Kierulf 於1987 年發展出來的，而後越來越普遍運用於圍棋棋譜中（Hollosi，1999）。SGF 檔案格式紀錄兩人盤面遊戲資訊，SGF 為一種文字格式，其提供了許多相關訊息，如：盤面的紀錄、說明、遊戲資訊... 等，為了將SGF 棋譜轉換到資料庫中，首先須對SGF 的內部架構有所了解，下圖為一個SGF檔案範例。

```
[;GM[1]AP[StoneBase(弈典);SGFParser.2.3]SZ[19]HA[0]KM[黑貼6目半]TM[30分]EV[新浪杯大學生圍棋賽]PB[吳奇]BR[6段]PW[陳亞虎]WR[6段]RE[黑中盤勝]DT[2005-01-01]PC[新浪圍棋];
B[qd];W[dc];B[dq];W[qp];B[ce];W[ed];B[oq];W[mq];B[po];W[qo];B[pn];W[qm];B[qq];W[rq];B[qn];W[rn];B[pp];W[rl];B[rr];W[qr];B[pq];W[rp];B[jq];W[mo];B[lp];W[mp];B[pm];W[pl];B[ol];W[nm];B[pk];W[ql];B[om];W[kn];B[in];W[kp];B[kq];W[lq];B[jp];W[lo];B[qi];W[do];B[fp];W[cq];B[cr];W[cp];B[dl];W[br];B[dr];W[fo];B[go];W[fn];B[cn];W[ep];B[fq];W[ch];B[fl];W[gn];B[ho];W[dn];B[dg];W[ei];B[cg];W[cm];B[cl];W[bl];B[bm];W[dm];B[bk];W[bn];B[al];W[cd];B[eh];W[fi];B[di];W[dj];B[fh];W[gi];B[ci];W[ek];B[el];W[gh];B[nc];W[jc];B[id];W[ic];B[gd];W[gf];B[hc];W[qc];B[pc];W[pd];B[qe];W[oc];B[pb];W[jd];B[ie];W[kf];B[hg];W[gg];B[jg];W[kg];B[jh];W[pg];B[pe];W[ok];B[nl];W[li];B[fe];W[ef];B[jj];W[ii];B[kh];W[lh];B[kj];W[hk];B[lc];W[jf];B[if])
```

圖表 2 SGF 檔案內容之舉例

SGF可分成兩大部分，第一部分為棋譜的賽事資訊。在這一部分記錄所有有關此次對弈的資訊，由一組英文簡寫記錄資訊的種類，[]中括號內則是記錄該種類的資料。如圖表 2:

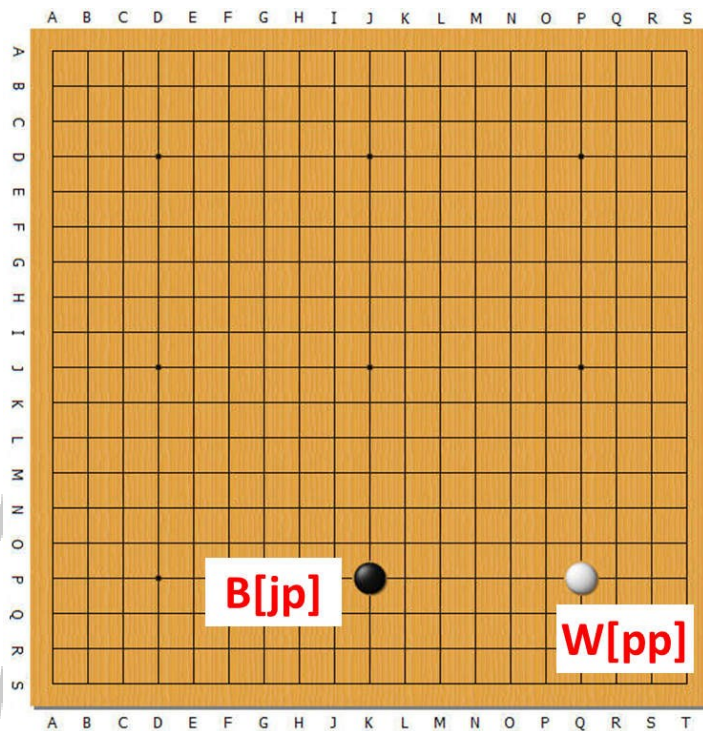
AP	打譜用的程式	BR	值黑者段數
SZ	棋盤大小	PW	值白手
KM	KM 貼目目數	WR	值白者段數
TM	Time 思考時間	RE	投降狀態
EV	賽事名稱	DT	Date 日期
PB	值黑手	PC	比賽場地

圖表 3 SGF 棋譜內容縮寫例解

SGF 的第二部分由一連串著手記錄組成，每一著手表示如下：

- 第一個字元為 B 或 W:分別代表黑子或白子中括號 [] 之中的二個小寫英文字母代表棋子位置，第一個小寫字母代表 x 軸座標，從左向右依序是 a、b、...、s 共 19 個

- 第二個小寫字母代表 y 軸座標，從上到下依序是 a、b、...、s 共 19 個。
例如， B[jp]以及 W[pp]分別的位置表示如圖表 4 之中的兩個棋子：

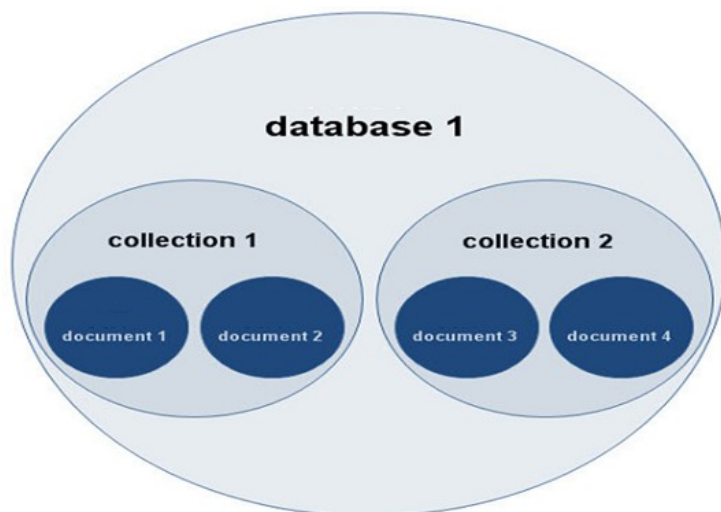


圖表 4 SGF 位置表示範例

2.2 MongoDB 資料庫

MongoDB 是由 10gen 團隊於 2007 年 10 月所創立，是一種文件導向資料庫，一款介於關聯式資料庫與非關聯式資料庫之間的產品。MongoDB 在功能與複雜度上取得了平衡，它擁有現今網路應用程式大部分所需要的功能：索引、複製、切割、豐富多樣查詢語法以及彈性的資料模型，而這些功能不會影響到速度。它結合了水平擴充以及關聯式資料庫常用的功能，如第二索引、範圍查詢及排序，也內建了許多常用的功能，如支援 MapReduce 型式的聚集和地理資訊的索引。

文件是(Document)mongoDB 資料的基本單位，大致與關聯式資料庫中的”列”(row)類似。類比來說，無網要資料庫的集合(Collection)就如同一張表格。一個單一 MongoDB 的實體可以管理多個獨立的資料庫(Database)，每個資料庫都可以擁有自己的集合以及權限。而每個文件都有一個叫做”Object_id”的特別鍵，它在整個文件的集合中是獨一無二的。



圖表 5 mongoDB 資料模型

關聯式資料庫(RDBMS)與 MongoDB 雖然在儲存概念相近，但在術語上用法是不同的。RDBMS 裡的 table，在 MongoDB 則稱為 Collection。Row 則是對應到 Document，Column 對應到 Field，Primary key 為 Object_id。

RDBMS	MongoDB
Table	Collection
Record/Row	Document
Column	Field
Primary Key	Object_id

圖表 6 關連式資料庫與 MongoDB 使用術語差異

MongoDB 裡面每一筆紀錄就是一個所謂的文件(Document)，文件導向式能讓你夠用單一筆記錄來表示複雜階層的關係。而最基本的概念“文件”是一個有順序的鍵值序列(key/value)。文件是 MongoDB 中資料的基本單位，大致上和關連式資料庫管理系統中的“列”(Row)類似，但文件又富含更多意義。像這個簡單的文件包含一個鍵，並且有一個值

```
{“Greetings”: “How are you?”}
```

這邊可以理解成在 Greetings 這個欄位裡，有一個“How are you?”的值。多數的文件會比這個還要複雜，並且會包含多個鍵值對，各鍵值對之間以逗號區隔，例如：

```
{“Greetings”:”How are you?,”foo”:3,”zoo”,”Tiger”}
```

在 MongoDB 的文件是以 JSON 格式來儲存，概念上如同 JavaScript 中的物件。JSON 式資料的簡單呈現法:它的規格可以只用一段文章就敘述完，能夠簡單的被理解、分析及記住，但 JSON 的表達能力較受限制，僅支援六種資

料型態:空字元(null)、布林(boolean)、數字(numeric)、字串(string)、陣列(array)及物件(object)。這些資料型態也許能夠表達大多數應用，但仍有一些額外的資料型態在資料庫中是相當重要的。例如，JSON 並沒有日期型態，這會讓與日期相關的格式混亂。JSON 雖然也有數字型態，但無法精準的辨認浮點數與整數，更無法區分 32 位元與 64 位元數字的差異。其他常用的像是，正規表示法或是函數，也無法呈現。因此 MongoDB 增加了一些額外的資料型態，例如時間、正規表示法、32/64 位元數字，但同時保持 JSON 必要的鍵/值對。每種資料型態如何被確切呈現是根據程式語言所決定。

<p>null :</p> <p>null 類型用於表示空值或不存在的欄位</p> <p>如：{"one":null}</p>	<p>Array:</p> <p>文檔中鍵值可以表示為陣列，在陣列內還可以嵌套陣列；如：</p> <p>{“x”:[“a”,“b”,[“c”,“d”]]}</p>
<p>boolean :</p> <p>布林類型有兩種值，'true'和'false'</p> <p>如：{"one":true}</p>	<p>內嵌 document :</p> <p>document 可以包含別的 document</p> <p>如：{"x":{"name":“Tom”,“age”:20}}</p>
<p>32 位元整數 :</p> <p>mongoDB 的控制台使用 JS 引擎進行輸入，而 JS 僅支援 64 位浮點數，所以 32 位元整數將會被自動轉義。</p>	<p>code : 文檔中可以包含 JS code</p> <p>如：{"one":function(){/*.....*/}}</p>
<p>64 位元整數 :</p> <p>64 位元整數與 32 位元整數一樣，在 MongoDB 控制台使用時，會轉義成 64 位浮點數</p>	<p>字符串 :</p> <p>UTF-8 字串都可以表示為字串類型的資料。</p> <p>如：{"one":“Hello World”}</p>
<p>64 位浮點數 :</p> <p>MongoDB 控制台數位的預設類型。</p> <p>如：{"one":2.02} {"one":10}</p>	<p>ObjectId 類型:</p> <p>對象 id 是文檔中唯一的 12 位的 ID</p> <p>0 1 2 3 4 5 6 7 8 9 10 11</p> <p>時間戳記 機器 PID 計數器，如：</p> <p>ObjectId("4eae239f63520362e051e7fd")</p>
<p>日期 :</p> <p>注意：使用的時候要加上 new</p> <p>如：{"one":new Date()}</p>	<p>正則運算式 :</p> <p>文檔鍵值可以包含規則運算式，其規則運算式採用 JS 語法來表示。</p> <p>如：{"one":/ho/i}</p>

圖表 7 MongoDB 結構

因為在 MongoDB 中文件被大量的使用，所以必須有個共通的文件表達方式在所有驅動程式、工具中共享，這種表式方式稱作二元進位的 JSON(Binary JSON)，又稱為 BSON。BSON 是一個輕量化的二元進位格式，它能夠用位元組的字串來表示所有的 MongoDB 文件。當某個驅動程式要求新增、查詢或是插入一個文件時，他會在文件被送至伺服器之前，將該文件編碼轉為 BSON 格式。文件從伺服器回傳回客戶端亦是使用 BSON 字串，會被驅動程式解碼為它的文件格式。

使用 BSON 格式主要有下列目的：

- (1)效率。BSON 是為了快速呈現資料而設計的，不需使用太多的空間，在儲存二元資料或是大型數據時，它會非常的有效率。
- (2) 移動性佳。在某些狀況，BSON 犧牲了一些空間效率，讓該格式能夠簡單的被移動，保留完整性，當 MongoDB 需要使用文件時，移動性是非常有用的。
- (3)效能。BSON 是使用 C 語言來表達，大多數的程式語言中都能夠被快速的處理。

Json	Bson
由 Douglas CroclFord 所提出	由 MongoDB 提出
全名 JavaScript Object Notation	全名 Binary JSON
是一個純文字的 LightWeight Data Format	基本上是以二進位的格式來表達 JSON 資料
以 JavaScript 為基礎，在加上物件導向的部份擴充特性	支援 JSON 不允許的 Date、RegEX 與二進位資料

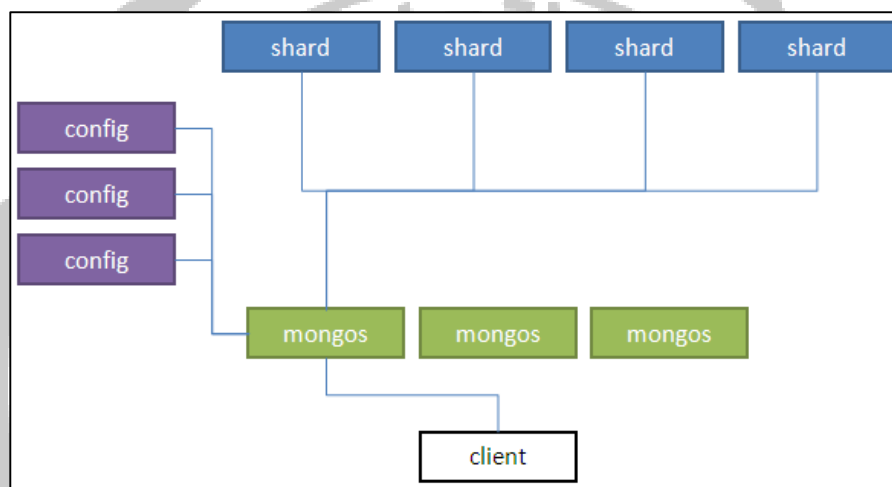
圖表 8 Jason 與 Bson 的差異比較

MongoDB 中，Sharding 的角色為把資料水平切分到不同的物理節點，利用更多的硬體資源來解決了單機性能極限的問題，數據量超過伺服器的硬碟容量時，就必需做 sharding。資料水平切分後，會減小每個索引的體積。索引一般都是 B 樹結構，索引體積減小後，索引深度也會隨之減小，索引查詢的速度也會隨之提高。Mongo 的 sharding 可以動態擴展、自動平衡、統一接口，搭配 replica set 提高可用性及容錯。Mongos，對用戶端來說，直接訪問的是 Mongos。它對外的介面就和普通的 mongod 一樣，可以使用標準 mongodb 用戶端和驅動進行訪問，主要作用是資料路由，定位資料位置，合併查詢結果。mongos 節點還負責資料移轉和資料自動平衡，並作為 sharding 集群的管理節點，不保存任何資料，可以任意水平擴展，這樣任意一個節點發生故障都可以很容易的進行容錯移轉。

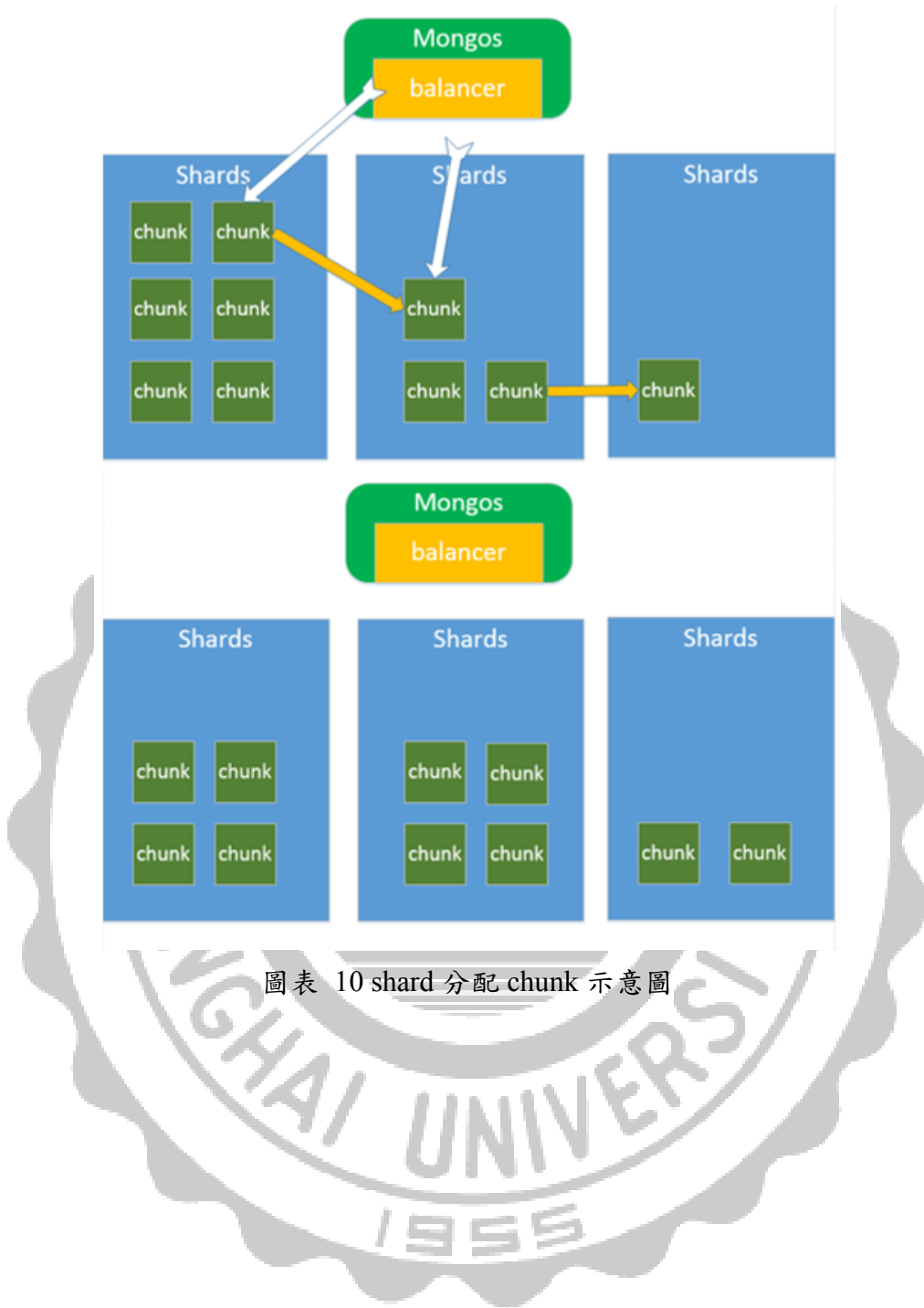
Config 在節點存儲了中繼資料，包括資料的位置，即哪些資料位於哪些節點，以及集群配置信息。Config 的節點也是普通的 mongod，這 3 個 config

節點並非是一個 replica set。它們的資料同步是由 mongos 執行兩階段提交來保證的。config 節點一定程度上實現了高可用。在一個或兩個節點發生故障時，config 集群會變成唯讀。但此時，整個 sharding 集群仍然可以正常讀寫資料。只是無法進行資料移轉和自動均衡而已。

Shard 則是實際存放資料的資料節點。每個 shard 節點可以是單個 mongod 實例，也可以是一個 replica set。通常在使用 sharding 的時候，都會同時使用 replica set 來實現高可用，避免單點故障的時候影響服務，及數據丟失。對於每個開啟 sharding 的 db 來說，都會有一個預設 shard。初始時，第一個 chunk 就會在那裡建立。新資料也就會先插入到那個 shard 節點中去。



圖表 9 sharding 架構圖



圖表 10 shard 分配 chunk 示意圖

第三章、系統架構與設計

3.1 系統架構

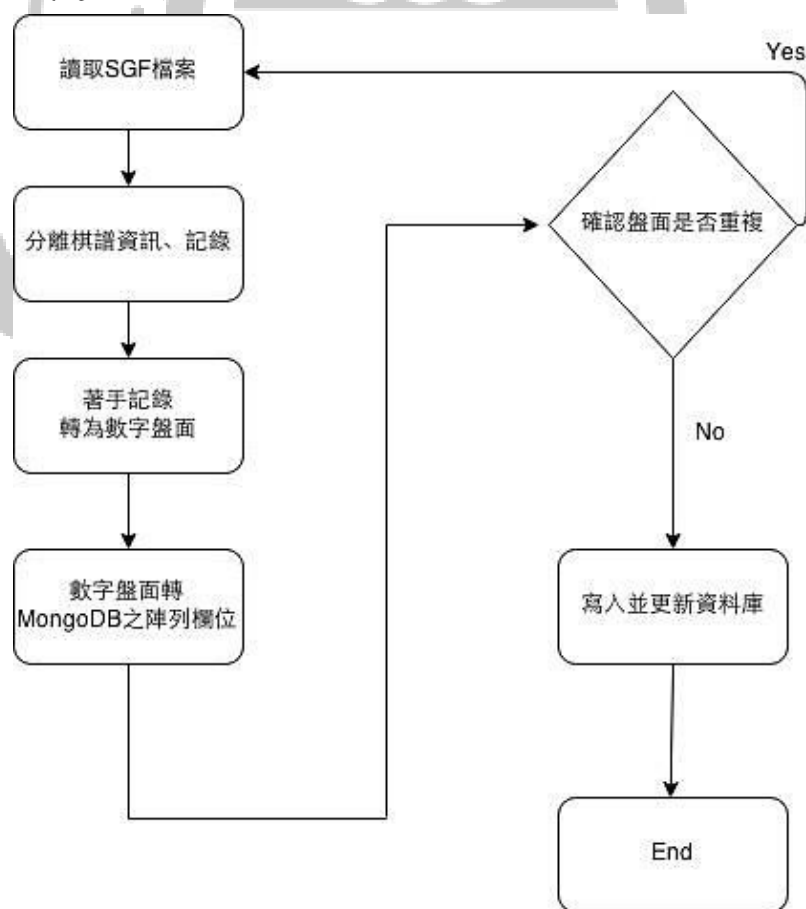
本章將介紹電腦圍棋棋譜搜尋系統架構。整個系統架構所示，可分為三個部分：

1. **棋譜擷取模組**：擷取圍棋網頁之棋譜 SGF 資料，分析取出棋譜資訊。
2. **資料轉換與儲存模組**：此模組把一個完整的圍棋盤面切割成所有可能的位移和旋轉的小盤面，然後儲存至資料庫。
3. **棋譜搜尋模組**：此模組讓使用者透過圖形化介面輸入要盤面樣式。此模組進行資料庫搜尋找出所有符合條件的盤面，並顯示在圖形化介面中。

以下三小節分別說明三個模組的運作流程。

3.2 SGF 棋譜擷取

棋譜網頁 www.weiqiok.com 蒐集了完整的棋譜資訊，我們採用HTTrack工具程式來定時擷取該網站。所擷取的網站資料為SGF格式。圖表11顯示SGF圍棋棋譜格式處理流程。



圖表 11 SGF 檔案分析處理流程圖

<pre>(;GM[1]AP[StoneBase(弈典):SGFParser.2.3]SZ[19]HA[0]KM[黑貼6目半]TM[30分]EV[新浪杯大學生圍棋賽]PB[吳奇]BR[6段]PW[陳亞虎]WR[6段]RE[黑中盤勝]DT[2005-01-01]PC[新浪圍棋];</pre>	棋譜賽事 資訊
<pre>B[qd];W[dc];B[dq];W[qp];B[ce];W[ed];B[oq];W[mq];B[po];W[qo];B[pn];W[qm];B[qq];W[rq];B[qn];W[rn];B[pp];W[rl];B[rr];W[qr];B[pq];W[rp];B[jc];W[mo];B[lp];W[mp];B[pm];W[pl];B[ol];W[nm];B[pk];W[ql];B[om];W[kn];B[in];W[kp];B[kq];W[lq];B[jp];W[lo];B[qi];W[do];B[fp];W[cq];B[cr];W[cp];B[dl];W[br];B[dr];W[fo];B[go];W[fn];B[cn];W[ep];B[fq];W[ch];B[fl];W[gn];B[ho];W[dn];B[dg];W[ei];B[cg];W[cm];B[cl];W[bl];B[bm];W[dm];B[bk];W[bn];B[al];W[cd];B[eh];W[fi];B[di];W[dj];B[fh];W[gi];B[ci];W[ek];B[el];W[h];B[nc];W[jc];B[id];W[ic];B[gd];W[gf];B[hc];W[qc];B[pc];W[pd];B[qe];W[oc];B[pb];W[jd];B[ie];W[kf];B[hg];W[gg];B[jg];W[kg];B[jh];W[pg];B[pe];W[ok];B[nl];W[li];B[fe];W[ef];B[jj];W[ii];B[kh];W[lh];B[kj];W[hk];B[lc];W[jf];B[if])</pre>	著手 順序&位置

圖表 12 SGF 棋譜賽事與著手資訊

每一個SGF格式檔案分別代表一個完整的圍棋棋譜，觀察圖表12，SGF檔案可分為兩個部分

- **賽事資訊**：SGF的賽事訊息會儲存於第一個和第二個分號之間。而賽事的詳細訊息，依據各縮寫而判斷再儲存於table內。
- **著手**：存在SGF的二個分號之後，以“;”隔開。

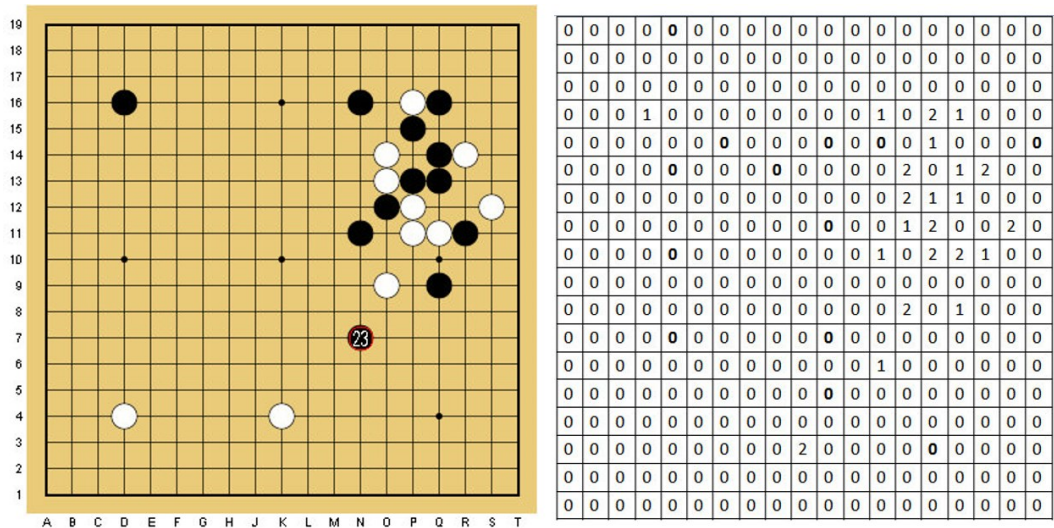
我們可以發現SGF有著一個規律性，那就是在記錄完第一段資料後，會在後方加註一個分號(;)當作區隔。由於第一段開始和結束都由分號著記，我們便可判別SGF的賽事訊息會儲存於第一個和第二個分號之間。而賽事的詳細訊息，依據各縮寫而判斷再儲存於table內。完成分割步驟後會留下每一手以一大寫B或W代表顏色，中括號及兩小寫代表位置的訊息，逐一判別個組合為一個著手訊息。

3.3 棋形的轉換切割和導入資料庫

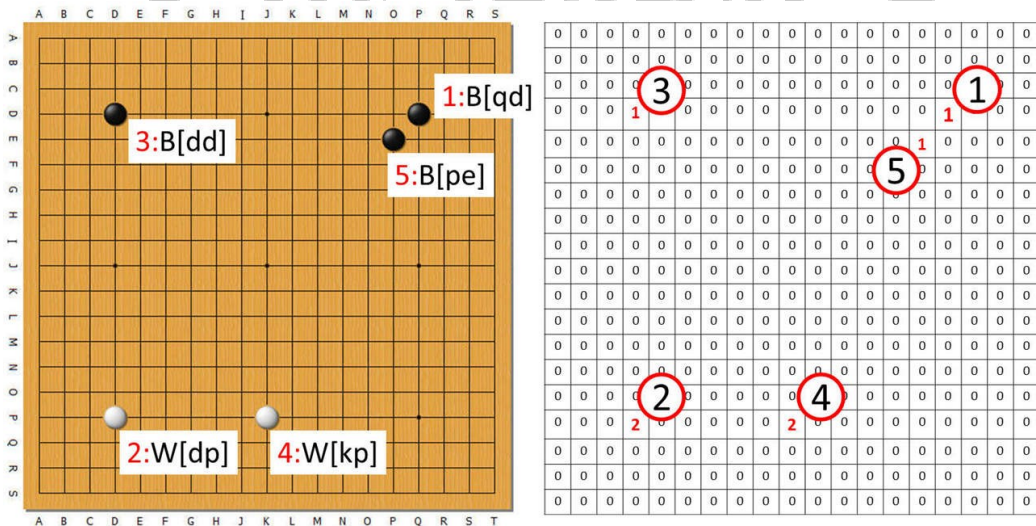
3.3.1 棋形轉換

將一場比賽的資訊存入資料庫時，該比賽的每一步著手都會被表示成一個19x19的數字盤面，其中每一元素有三種可能的數值：0代表空白、1代表黑子、2代表白子。

假設一場比賽有 N 步著手，每一步著手都會對應到一個數字盤面。最多會有N個 19x19 的數字盤面。圖表 14 和圖表 15 顯示前著手的數字盤面。



圖表 13 數字盤面示意圖

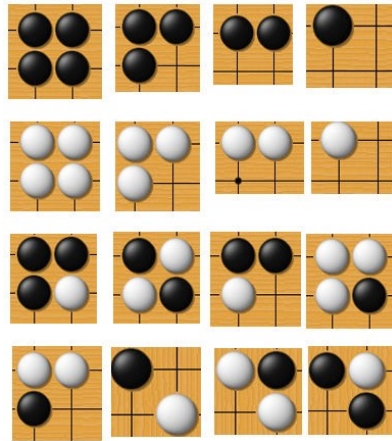


圖表 14 圖表 13 之前五步著手(B[qd];W[dp];B[dd];W[kp];B[pe];)示意圖

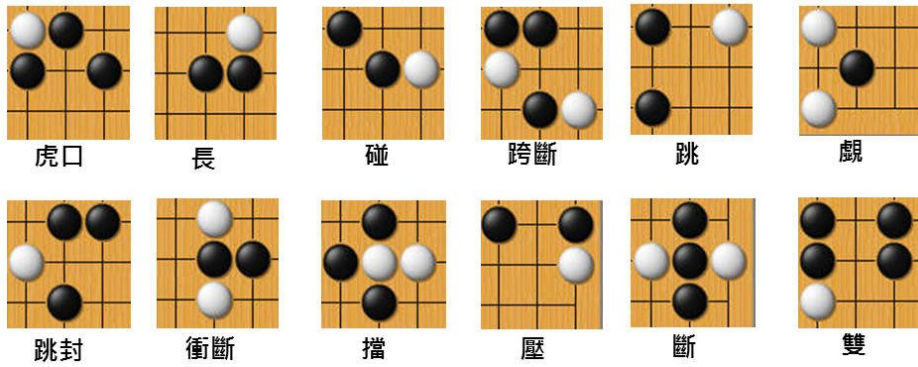
3.3.2 盤面切割

由於資料庫是以完全比對為基礎，我們將 19x19 的數字盤面切割成 3x3 的小的盤面來儲存。若是切割成小於 3X3 這個大小，重複性會太高，意義不大。例如圖表 15 中，2X2 棋形的變化只有 16(2⁴)種，缺乏識別度，而 3X3 棋形的變化卻多達 512(2⁸)。

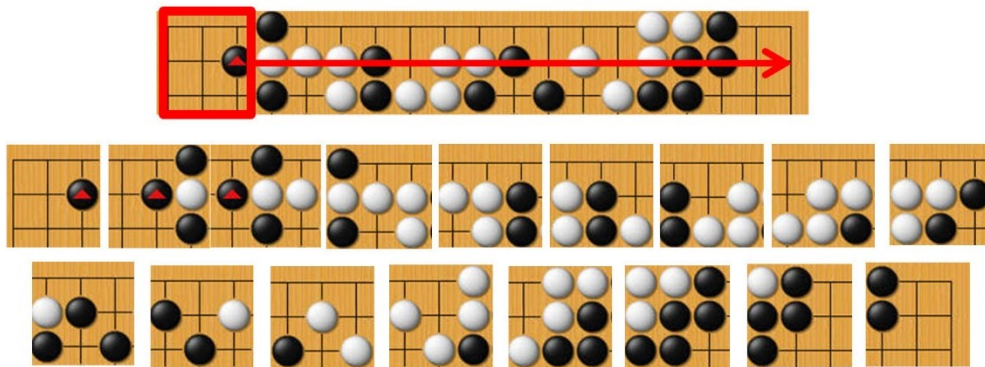
單就 3X3 大小而言，主要是讓電腦能處理簡單的棋形，最常辨認的有眼位，此類棋形的資料量較少，可讓電腦快速掃描以及更新。至於為何不使用 4X4、5X5 甚至是 6X6 格式的原因，則是因為格式越大會影響搜尋比對的速度，以及比對到相同棋形的機率也相對較低，且絕大多數的圍棋手段，例如長、跳、衝、斷等等，都能夠包含在 3X3 的範圍之中，故以 3X3 大小已足以適用於一般常用棋形。



圖表 15 2X2 棋形變化

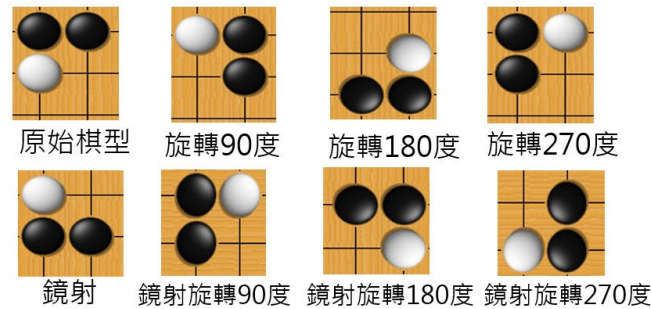


圖表 16 圍棋手段的常見棋形範例



圖表 17 3X3 棋形切割示意圖

棋形切割時，會從盤面左端向右掃描，如圖表 17，第一排會產生 17 個 3X3 棋形。我們考慮到棋形會有旋轉和鏡射共八種的狀況發生，如圖表 18，雖然乍看之下是不一樣的棋形，但事實上他們是相同的棋形。



圖表 18 棋形的旋轉角度與鏡射示意圖

旋轉和鏡射處理後的 3x3 棋形會被存入資料庫，作為搜尋的基本單位。系統並不會儲存所有的 3x3 棋形。當棋形全為空白時，則不予儲存。如此一來，資料庫的容量，可得到控管，不會過度浪費。

3.3.3 資料庫欄位

本論文使用 MongoDB 資料庫來儲存資料，MongoDB 為 NoSQL 資料庫，NoSQL 在資料儲存時為一個 Key 值對應其 Value，並不像 RDMS 能夠在不同 table 間互相聯結，所以在設計資料庫欄位是必須精準而且切割詳細。

在 MongoDB 資料庫中建立一個 go 資料庫，內含 1 個集合(Collection)名稱，為 gocollection。

gocollection 包含了以下欄位：

- Event: SGF 棋譜檔案的賽事名稱
- Size: SGF 棋譜的棋盤大小
- KM: 該賽事的貼目數
- Date: 賽事日期
- Step: 每一步著手數
- Co: Coordinate，每一個切割的盤面會賦予一個座標，供系統利用
- SGF: 儲存原始的 SGF 檔案字串。
- _id: 為 Primary key，不可重複，用來辨識該棋盤與棋形是否相同
- PlayerBlack: 執黑子玩家姓名
- PlayerWhite: 執白子玩家姓名
- Resign: 認輸的玩家
- Place: 賽事場地
- FullBoard: 該著手數的完整盤面，大小為 19X19
- Spboard: 分割成 3X3 大小的小盤面，為陣列，內有 2312(17X17X8) 筆資料

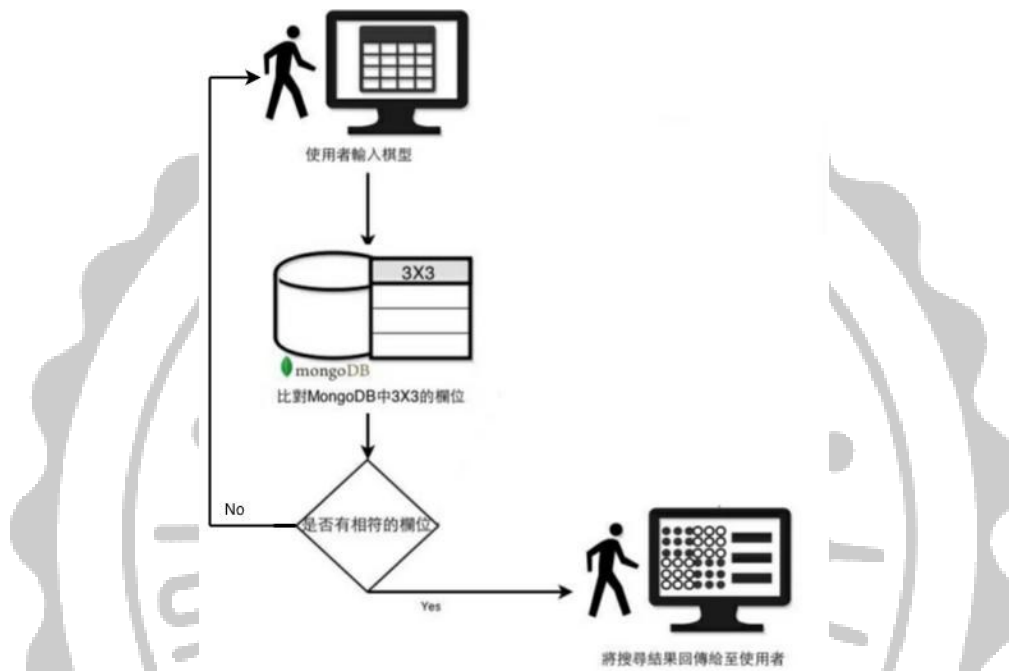
圖表 19 gotable 棋譜資料庫資料型態表顯示了 gotable 的欄位以及資料型態。

欄位	Event	Size	KM	Date	Step	Fullboard	Co	spboard
資料型態	varchar	int	Int	varchar	int	varchar	int	varchar

欄位	SGF	_id	PlayerBlack	PlayerWhite	Resign	Place	FullBoard
資料型態	varchar	varchar	varchar	varchar	varchar	varchar	varchar

圖表 19 gotable 棋譜資料庫資料型態表

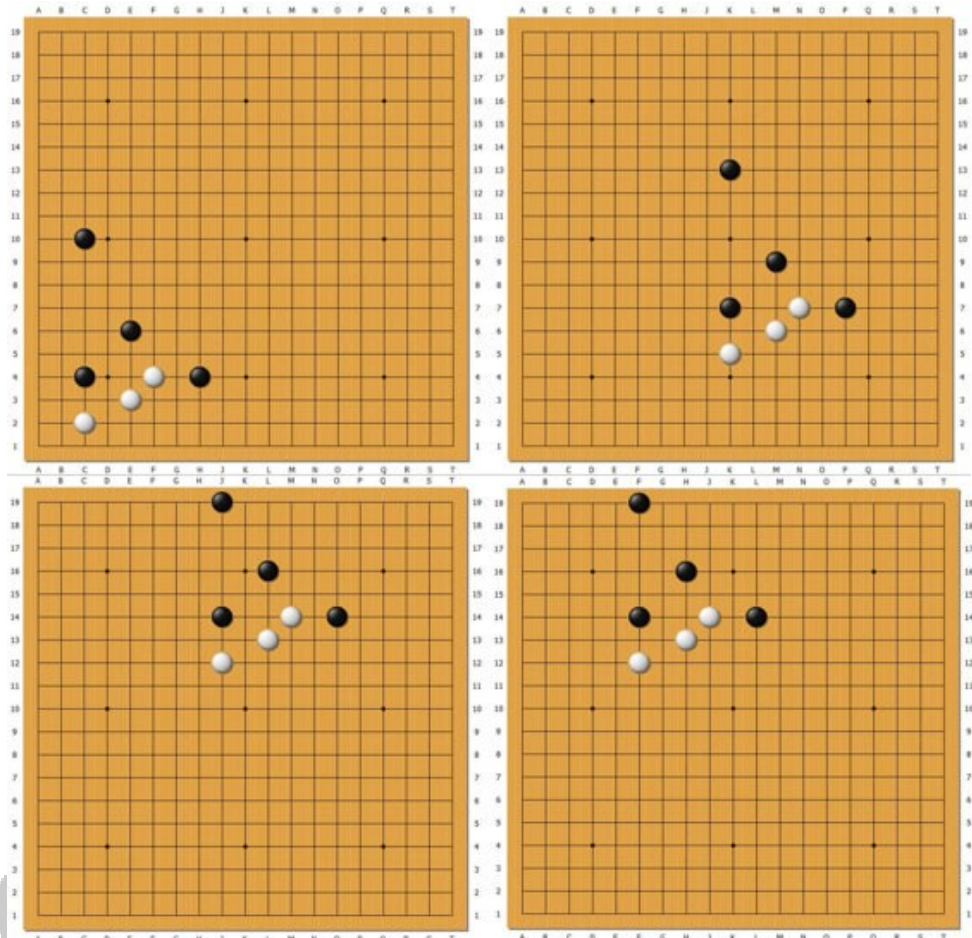
3.3 棋形搜尋



圖表 20 棋形搜尋流程圖

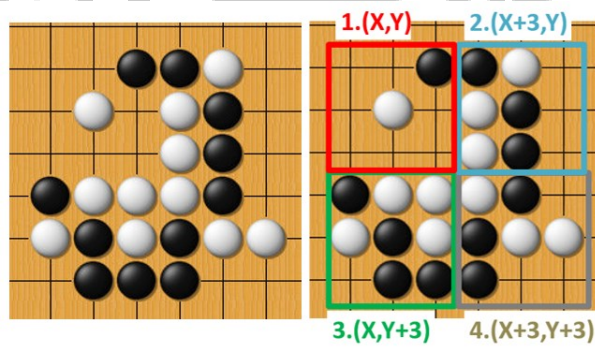
當系統執行棋形搜尋時，系統會先要求使用者輸入欲搜尋的棋形，輸入完畢後，便會開始在mongoDB中比對棋形的資料欄位，而過程如下：

1. 要求使用者在搜尋介面輸入欲搜尋的棋形，這個步驟會列出完整的19X19棋盤供使用者輸入棋形，使用者可在任意座標輸入棋形，在不同座標輸入的棋形，依然可搜尋到相同結果的棋形，如圖表 21相異座標但相同棋形之示意圖圖表 20。
2. 搜尋系統連結至mongoDB，並逐項比對mongoDB內的陣列欄位中每一個盤面。
3. 若該盤面與搜尋棋形相符，則列出搜尋結果，但是此狀況相當罕見，除非使用者輸入的棋形相當簡單，這個狀況通常只會在比賽剛開局的前幾手時出現。
4. 將搜尋結果回傳至使用者的介面

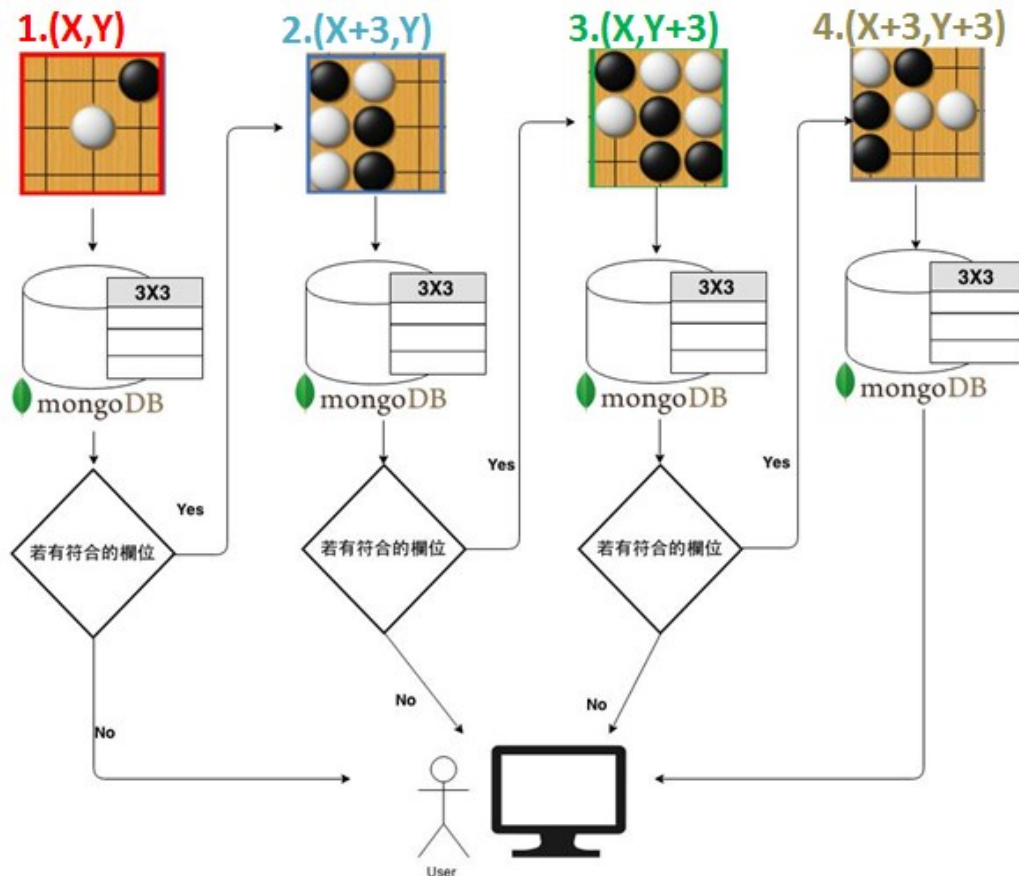


圖表 21 相異座標但相同棋形之示意圖

當使用者輸入完欲搜尋的棋形時，系統會針對該棋形自動以3X3大小的框架切割該棋形，分配相對座標與編號，如圖表 22。



圖表 22 棋形搜尋的分割與座標分配示意圖



圖表 23 3X3 分割棋形的搜尋流程圖

如圖表 23 3X3 分割棋形的搜尋流程圖 圖表 22，系統搜尋時，會依序從編號 1 的棋形開始比對 mongoDB 中 3X3 大小的棋形欄位，若有搜尋到符合的欄位，會使用下一編號的棋形之相對座標，以編號 2 的棋形繼續比對 mongoDB 中的棋形資料欄位，系統會依續搜尋至編號末端，以此組合分割過的棋形。遇到無法找到的狀況時，則會停止搜尋，並回傳結果至使用者介面。

第四章、系統實作

在本章中，將對本系統（圍棋棋譜棋形比對系統），一共分成三個部分，分別是 SGF 檔案分析處理程式、導入 MongoDB 資料庫程式以及棋形搜尋程式。

4.1 SGF 檔案分析處理程式

本論文所擷取的棋譜網頁為：圍棋學研網 weiqiok.com，收藏的賽事是以中國、日本以及韓國各大賽事為基礎的網頁。

- 讀取SGF檔

```
public static void main(String[] args) throws IOException {
    Path rootD = Paths.get("c:\\shiju1");
    //取得要掃描的目錄
    Files.walkFileTree(rootD, new Clone.FindsgfVisitor());
    public static class FindsgfVisitor extends SimpleFileVisitor<Path> {
        int i = 1;
        private BufferedReader reader = null;
        public FileVisitResult preVisitDirectory(Path dir, BasicFileAttributes attrs) {{
            i = 1;
            System.out.println("目錄: " + dir); }
        return FileVisitResult.CONTINUE; //繼續目錄下層找}
    }
    String s = line; //讀入SGF檔
    StringTokenizer st = new
    StringTokenizer(s, ";") //將SGF檔以分號切割
    while (st.hasMoreTokens()) // 將切割後的資料存入mongoDB
```

- 分離棋譜資訊、記錄

```
public FileVisitResult visitFile(Path file, BasicFileAttributes attrs) throws IOException
{//掃描檔案
int countstone = 0;
//掃描檔案
while (file.toString().endsWith(".sgf")) {
    //如果檔案結尾為.sgf，讀入程式
    System.out.println(i + file.getFileName());
    //取得檔案名稱並給予編號
    i++;
    Files.readAllBytes(file);
    for (String line : Files.readAllLines(file)) {
        System.out.println(line);
    }
}
```

```

String in2 = line;
String s2 = line;
StringTokenizer st2 = new StringTokenizer(s2, ",");
System.out.println("\n=====");
    //取出st2的內容，用「,」來分解著手資訊
System.out.println(s2 + "字串拆解為:");
while (st2.hasMoreTokens()) {
    String step = st2.nextToken();
    MongoClient mongo = new MongoClient("localhost", 27017);
    DB db = mongo.getDB("test55");
    DBCollection table = db.getCollection("go55");
    BasicDBObject document = new BasicDBObject();
    Pattern p2 = Pattern.compile("EV\\[(.*?)\\]");
    Matcher m2 = p2.matcher(in2);
    boolean matchFound2 = m2.find();
    //分割賽事資訊，ex.對弈者資訊、場地、時間

```

- 將SGF著手紀錄轉為MongoDB之陣列欄位並存入資料庫

```

public static int getPieceId(char p) //判斷黑子或白子
public static int getCellId(char x) //判斷該黑/白子的位置
public static int printBoard() //顯示判斷結果，並存入MongoDB

```

- 棋形切割: MongoDB 之陣列欄位切割並存入資料庫

```

for (a = 0; a < 19; a++) {
    for (b = 0; b < 19; b++) {
        //依序將每份棋譜的盤面由讀入
    }
    for (o = row_start; o <= row_end; o++) { //將盤面切割至 3X3 的大小
        for (n = col_start; n <= col_end; n++) {
            //儲存棋形
        }
    }
}

```

- 寫入棋譜賽事資訊並更新資料庫

```

MongoClient mongo = new MongoClient("localhost", 27017);
DB db = mongo.getDB("test55");
DBCollection table = db.getCollection("go55");
BasicDBObject document = new BasicDBObject();
//建立 document 元件，提供 JAVA 寫入 mongoDB
document.put("Event", groupStr2); //賽事名稱，document 元件
document.put("Size", groupStr2) //比賽用棋盤大小，document 元件
document.put("KM", groupStr2); //貼目數，document 元件

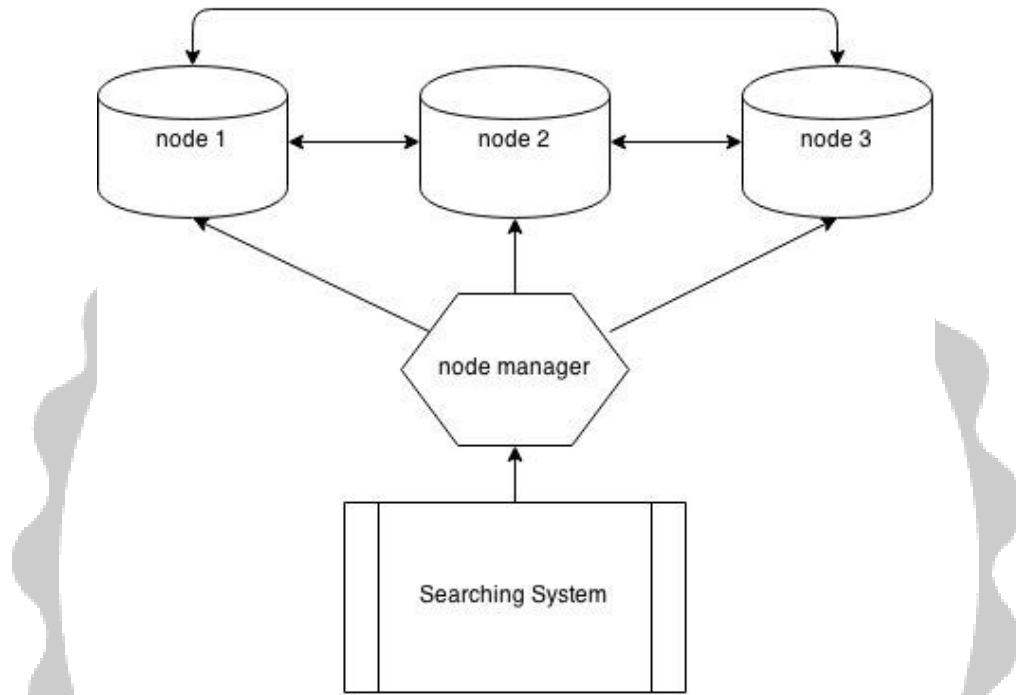
```

```

document.put("PlayerBlack", groupStr2); //執黑子玩家姓名， document 元件
document.put("PlayerWhite", groupStr2); //執黑子玩家姓名， document 元件
document.put("Place", groupStr2); //賽事地點， 寫入 document 元件
document.put("Step", countstone); //下棋順序數， 寫入 document 元件
table.insert(document); //將 document 內佇列資料， 寫入 mongodb

```

4.2 導入 MongoDB 之程式



圖表 24 棋譜搜尋系統資料庫架構圖

如圖表 24，本論文的資料庫使用的系統為 MongoDB，MongoDB 預設一個基本的管理者，用來管理水平切割的資料庫。為了加快搜尋速度，本系統分割為三個子節點，一共準備了三部主機，在三台主機各自安裝 MongoDB server。其 IP 分別是 192.168.1.2 (主機名稱 node1)、192.168.1.3 (主機名稱 node2) 與 192.168.1.4 (主機名稱 node3)，其中 192.168.1.2 與 192.168.1.3 分別是標準節點，而 192.168.1.4 則是被動節點，每個節點互相連接。

在初始化之前，必須確保每個節點的 port: 27017 是可以互通不被防火牆擋掉的假設我們一開始以 192.168.1.2 做為 Primary，我們就在 192.168.1.2 的機器上，執行 cmd.exe，並透過 mongo 這個指令連進服務。

```

Microsoft Windows [版本 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

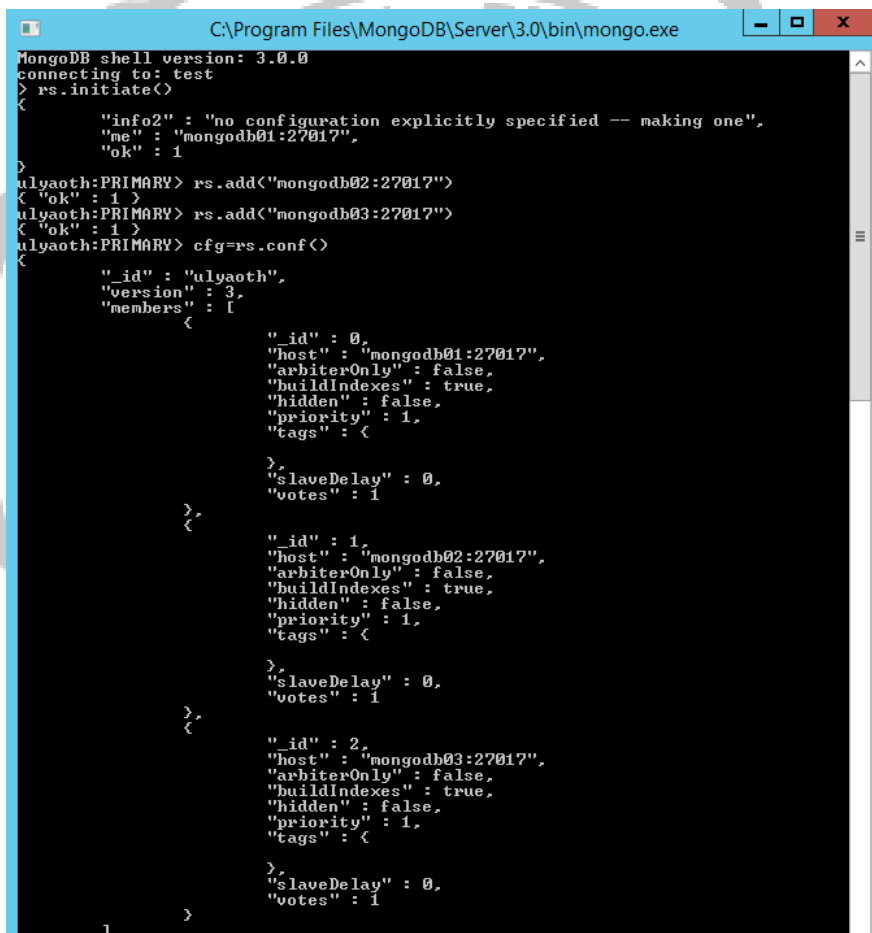
C:\Windows\system32>cd C:\mongodb-win32-x86_64-2008plus-2.4.5\bin
C:\mongodb-win32-x86_64-2008plus-2.4.5\bin>mongo
MongoDB shell version: 2.4.5
connecting to: test

```

圖表 25 mongoDB 初始化

以下圖初始化的指令:

```
rs.initiate()  
//初始化 replica set 服務  
rs.add("192.168.1.3")  
//將 192.168.1.3 加入至 replica set 服務  
rs.add("192.168.1.4")  
//將 192.168.1.4 加入至 replica set 服務  
rs.conf()  
//檢視設定  
rs.status()  
//檢視狀態
```



```
C:\Program Files\MongoDB\Server\3.0\bin\mongo.exe  
MongoDB shell version: 3.0.0  
connecting to: test  
> rs.initiate(<  
<  
  "info2" : "no configuration explicitly specified -- making one",  
  "me" : "mongodb01:27017",  
  "ok" : 1  
>  
ulyaoth:PRIMARY> rs.add("mongodb02:27017")  
< "ok" : 1 >  
ulyaoth:PRIMARY> rs.add("mongodb03:27017")  
< "ok" : 1 >  
ulyaoth:PRIMARY> cfg=rs.conf(<  
<  
  "_id" : "ulyaoth",  
  "version" : 3,  
  "members" : [  
    <  
      "_id" : 0,  
      "host" : "mongodb01:27017",  
      "arbiterOnly" : false,  
      "buildIndexes" : true,  
      "hidden" : false,  
      "priority" : 1,  
      "tags" : <  
    >,  
      "slaveDelay" : 0,  
      "votes" : 1  
    >,  
    <  
      "_id" : 1,  
      "host" : "mongodb02:27017",  
      "arbiterOnly" : false,  
      "buildIndexes" : true,  
      "hidden" : false,  
      "priority" : 1,  
      "tags" : <  
    >,  
      "slaveDelay" : 0,  
      "votes" : 1  
    >,  
    <  
      "_id" : 2,  
      "host" : "mongodb03:27017",  
      "arbiterOnly" : false,  
      "buildIndexes" : true,  
      "hidden" : false,  
      "priority" : 1,  
      "tags" : <  
    >,  
      "slaveDelay" : 0,  
      "votes" : 1  
    >  
  ]  
>  
>  
1.
```

圖表 26 mongoDB 初始化完成

如圖表 26，這樣MongoDB的初始化就算完成，便可開始使用MongoDB API來儲存棋譜資訊。

MongoDB API

MongoDB API為本系統使用，是一種JAVA函式，它的功能如下

- 建立一個JAVA編譯檔

```
mvn archetype:generate -DgroupId=com.goboard.core -DartifactId=mongodb
-DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
//建立以 maven 編譯器的 JAVA 檔案
```

- 連接MongoDB資料庫

```
MongoClient mongo = new MongoClient("localhost", 27017);
//連接MongoDB，27017連接到本地端
DB db = mongo.getDB("mongotest");
//連接資料庫mongotest，若不存在則自動建立
List<String> dbs = mongo.getDatabaseNames();
for(String db : dbs){
    System.out.println(db); }//列出所有的資料庫
```

- 建立MongoDB的加密連線

```
MongoClient mongoClient = new MongoClient();
DB db = mongoClient.getDB("mongotest");
boolean auth = db.authenticate("username", "password".toCharArray());
//如果 mongoDB 為加密模式，連接資料庫 mongotest 須輸入密碼
```

- 建立MongoDB集合

```
DBCollection table = db.getCollection("user");
//連接集合"user"如果集合名稱不存在，則自動建立
DB db = mongo.getDB("testdb");
Set<String> tables = db.getCollectionNames();
for(String coll : tables){
    System.out.println(coll);
}
//顯示所有的集合
```

- 將資料儲存至資料欄位

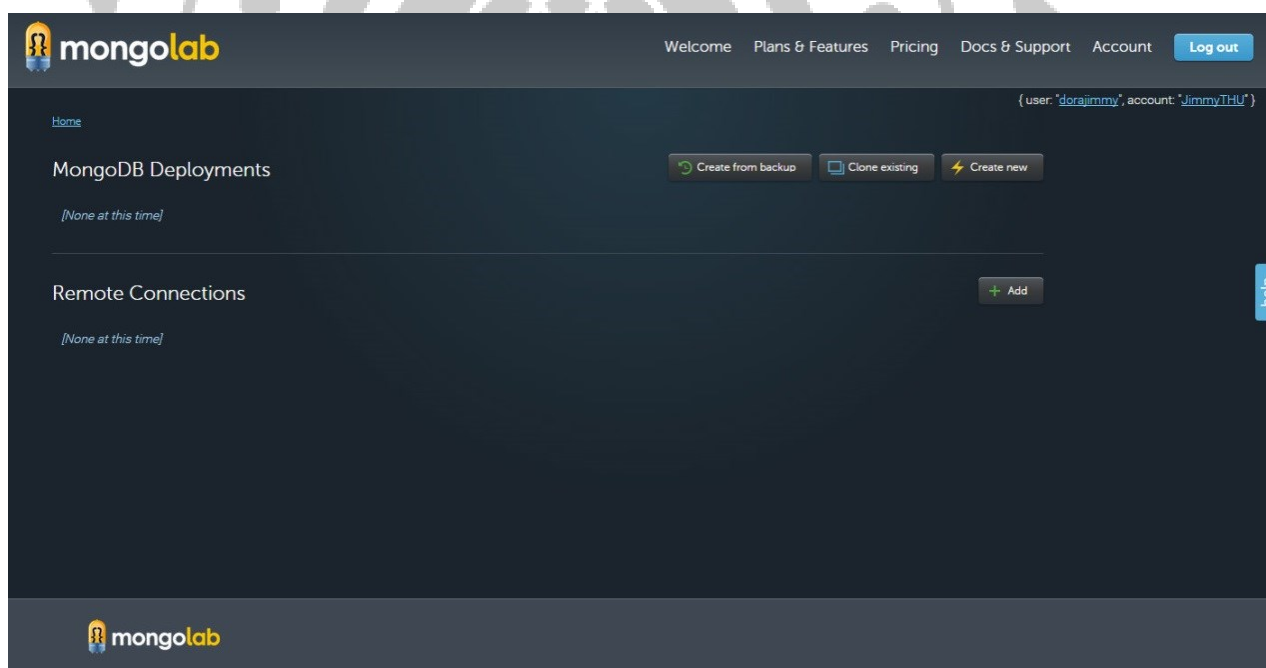
```
DBCollection table = db.getCollection("user");
BasicDBObject document = new BasicDBObject();
document.put("name", "gotest");//欄位名稱 name,內容 gotest
document.put("age", 10);//欄位名稱 age，內容 10
document.put("createDate", new Date());
//欄位名稱 createDate，內容為當日日期
table.insert(document);// 將佇列資料欄位存入資料庫
```

- 尋找一個欄位資料

```
DBCollection table = db.getCollection("user");
BasicDBObject searchQuery = new BasicDBObject();
searchQuery.put("EventNo.", "Go0123");
//尋找 EventNo.欄位，內容為 Go0123 的資料
DBCursor cursor = table.find(searchQuery);
while (cursor.hasNext()) {
    System.out.println(cursor.next());
} //當資料找到時，列出搜尋結果
```

- 查詢棋形

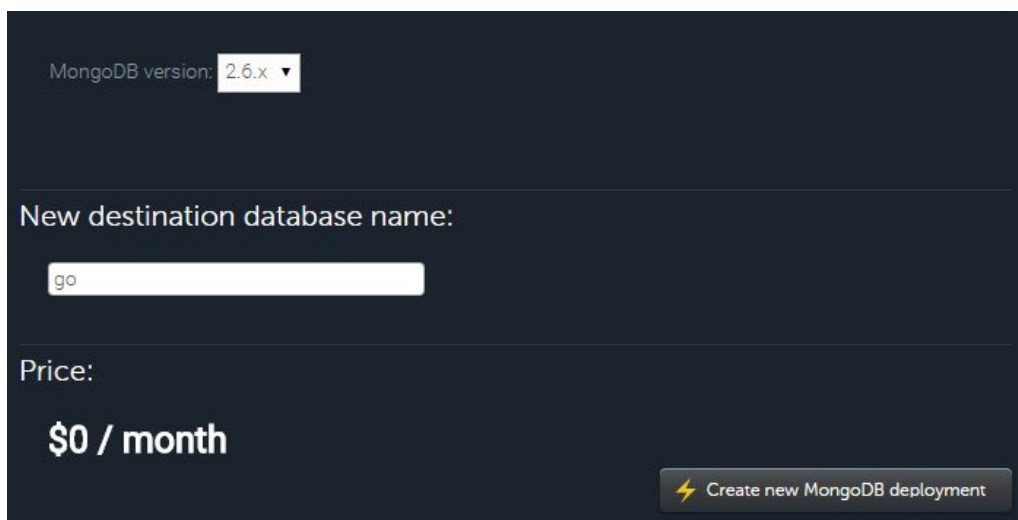
```
BasicDBObject searchQueryRow= new BasicDBObject(); //建立搜尋物件
searchQueryRow.put("row:", new BasicDBObject("$regex", re));
//將搜尋棋形置入搜尋物件，並切割欲搜尋的棋形
DBCursor cursorRow = table.find(searchQueryRow);
//將搜尋物件比對 Collection 中的文件
while (cursorRow.hasNext()) { //若找到棋形，回傳棋譜賽事資訊 }
```



圖表 27 MongLab 線上 MongoDB 服務畫面

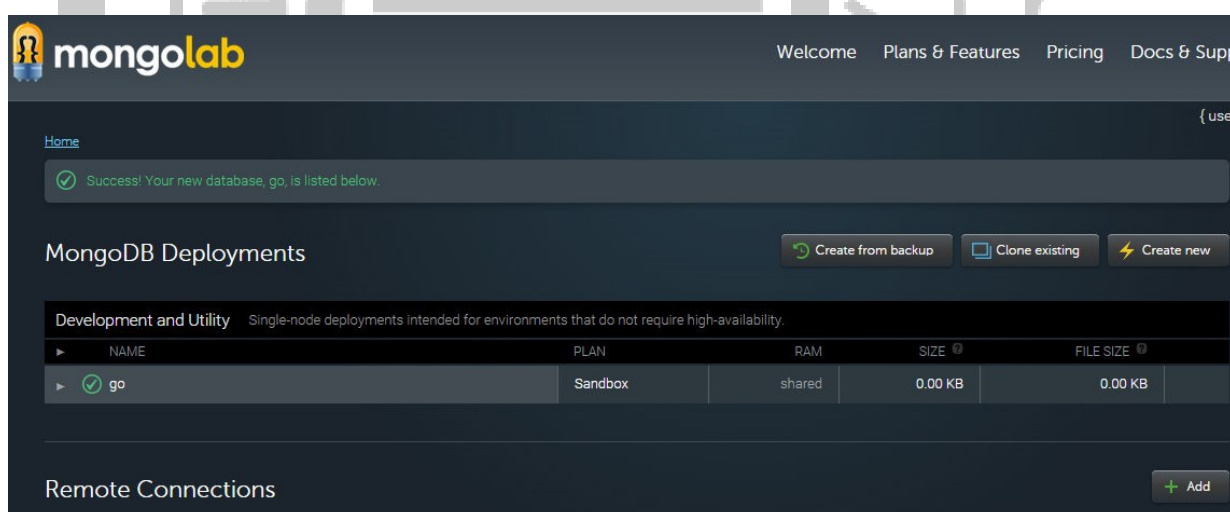
由於 MongoDB 本身並無圖形化界面供使用者利用，所以我們使用了 mongolab.com 這個線上 MongoDB server 來介紹棋譜資料是如何儲存和搜尋，如圖表 27，Mongolab 是以 Web 和圖形化介面提供 MongoDB server 的服務的網站。

以下以 Mongolab 範例將使用 2 筆 19X19 的著手記錄盤面，切割成 578 個 (2X17X17) 3X3 棋形存入 Mongolab 並進行 3X3 棋形查詢。



圖表 28 Mongolab 建立 database

如圖表 28，帳號建立完成，按下 Create new 按鈕，就可以準備建立 Database，按下最底下的 Create new MongoDB deployment 按鈕，就會根據設定填入的資料的開始建立 Database，完成之後會回到一開始的管理介面，如圖表 29，這裡可以看到使用者建立的所有資料庫，點入該資料庫名稱即可進行進階的管理動作。



圖表 29 Mongolab 管理界面

點入該資料庫名稱，即可 Collection 名稱建立 Collection，建立完成後，如圖表 30，可以看到使用者建立的所有 Collection，我們在這裡建立了儲存棋譜賽事資訊的 gocollection 以及儲存 19X19 盤面和 3X3 盤面的 SpiltBoard。

Home

Database: go Delete database

To connect using the shell:

```
% mongo ds031701.mongolab.com:31701/go -u <dbuser> -p <dbpassword>
```

To connect using a driver via the standard URI ([what's this?](#)):

```
mongodb://<dbuser>:<dbpassword>@ds031701.mongolab.com:31701/go
```

mongod 2.6.9

⚠ A database user is required to connect to this database. [Click here](#) to create a new one.

Collections
Users
Stats
Backups
Tools

Collections + Add collection

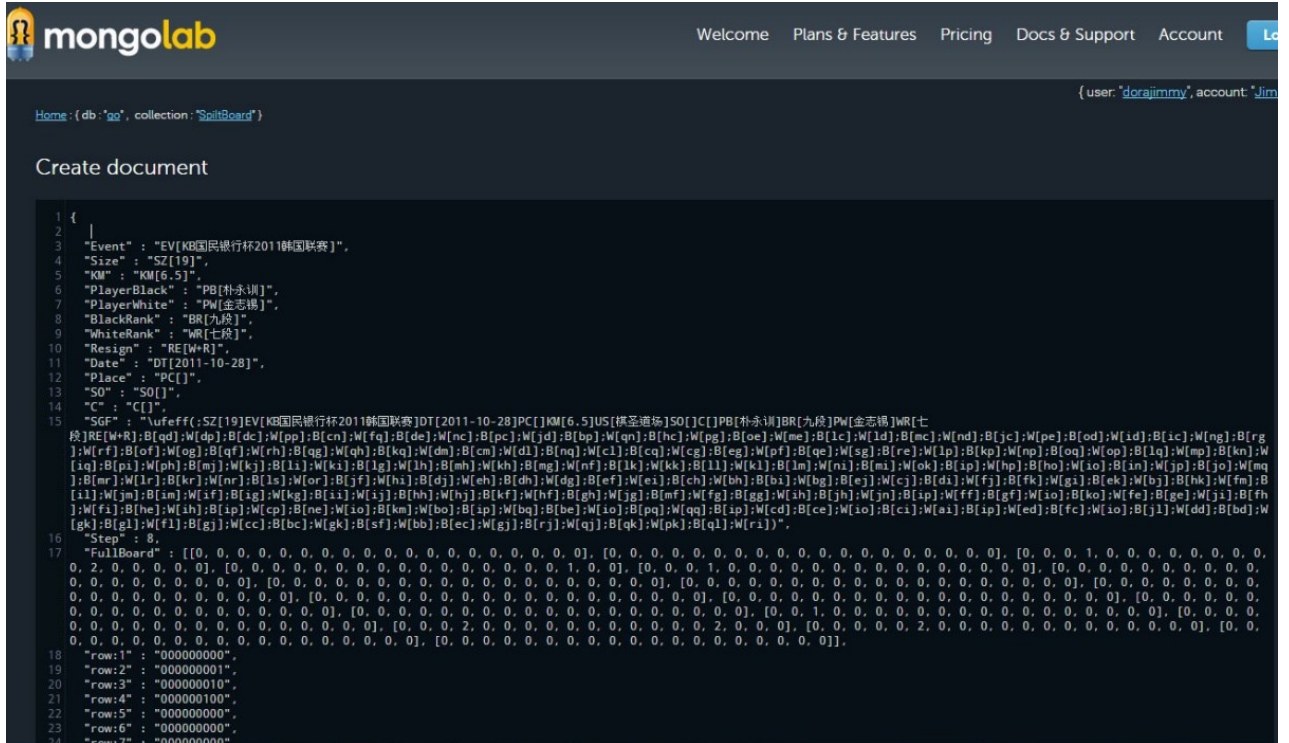
NAME	DOCUMENTS	CAPPED?	SIZE
gocollection	2	false	39.95 KB
SpiltBoard	578	false	71.92MB

System Collections

NAME	DOCUMENTS	SIZE
system.indexes	2	0.22 KB

圖表 30 Mongolab Collection 管理畫面

預設剛建立完成的 collection 內部是空的，如圖表 31，需點入欲建立 document 的 collection 名稱並輸入 Json 格式的內文。圖表 31 內為經由 JAVA 程式處理過的 3X3 棋譜資料。



圖表 31 Mongolab 建立 Document 畫面

建立好資料後，即可在 Mongolab 上對 Collection 進行查詢，如圖表 32，在中間的 Query (blank returns all objects) 方塊輸入查詢條件，再按下中間右邊的 Search 按鈕，就可以看到查詢的結果。

--- Start new search ---

Query (blank returns all objects):

```
{
  "row:1": "010010010"
}
```

Sort order (1: asc, -1: desc):

Subset of fields (1: include, 0: exclude):

Reset Save this search Search

Search Results

Display mode: list table ([edit table view](#))

records / page 10 [1 - 5 of 5]

<pre>{ "_id": { "\$oid": "556c561ce4b0c38aae33b4df" }, "Event": "EV[KB国民银行杯2011韩国联赛]", "Size": "SZ[19]", "KM": "KM[6 5]" }</pre>	<input type="button" value="x"/> <input type="button" value="↗"/>
<pre>{ "_id": { "\$oid": "556c5661e4b0c38aae33b4e1" }, "Event": "EV[KB国民银行杯2011韩国联赛]", "Size": "SZ[19]", "KM": "KM[6 5]" }</pre>	<input type="button" value="x"/> <input type="button" value="↗"/>
<pre>{ "_id": { "\$oid": "556c567be4b0c38aae33b4e2" }, "Event": "EV[KB国民银行杯2011韩国联赛]", "Size": "SZ[19]", "KM": "KM[6 5]" }</pre>	<input type="button" value="x"/> <input type="button" value="↗"/>
<pre>{ "_id": { "\$oid": "556c5688e4b0c38aae33b4e4" }, "Event": "EV[KB国民银行杯2011韩国联赛]", "Size": "SZ[19]", "KM": "KM[6 5]" }</pre>	<input type="button" value="x"/> <input type="button" value="↗"/>
<pre>{ "_id": { "\$oid": "556d1b3be4b0c38aae33cbe3" }, "Event": "EV[KB国民银行杯2011韩国联赛]", "Size": "SZ[19]", "KM": "KM[6 5]" }</pre>	<input type="button" value="x"/> <input type="button" value="↗"/>

records / page 10 [1 - 5 of 5]

圖表 32 Mongolab 查詢模型畫面

4.3 棋形搜尋程式



圖表 33 棋形搜尋程式操作流程



圖表 34 系統搜尋介面

本搜尋系統介面以JAVA撰寫，使用者在介面點擊後會連結MongoDB上去搜尋該棋形，並將結果列出來。Java驅動程式是最老的MongoDB驅動程式，他被使

用在產品已經相當多年，是個穩定又是企業開發者的人氣選擇。我們將會使用 Java 驅動程式來建立一個搜尋棋形的搜尋引擎。

使用者需在圍棋搜尋系統上使用 19X19 的盤面上點選欲搜尋棋形，預設第一手為黑色，點選乙次後變更為白色，依此規則輸入棋形。若欲連續輸入相同顏色的棋子，可在左方分別點選按鈕”B”，連續輸入黑子，點選”W”連續輸入白子。欲回到預設狀態擇點選”B/W”。輸入棋形完畢後，按下按鈕”Search”開始連結資料庫，並搜尋相對應棋形，搜尋結果則顯示於畫面上方，結果為該棋形之賽事相關資料。

4.4 棋形搜尋驗證

以下將驗證查詢到的棋形是否真的存在於 MongoDB 內當中。



圖表 35 MongoLab 新增查詢

如圖表 35 MongoLab 新增查詢，驗證將使用 MongoLab 內的查詢功能。



圖表 36 MongoLab 輸入搜尋畫面

如圖圖表 36 MongoLab 輸入搜尋畫面，在搜尋欄輸入欲搜尋的棋形，輸入完畢後按下 search 對 spboard 欄位進行棋型搜尋。

The screenshot shows the MongoDBLab interface for a collection named 'gocollection1'. The search query is:

```
{
  "spboard": [
    0,0,0,
    0,0,0,
    0,2,2
  ]
}
```

The search results are displayed in a list view, showing the first three records. Each record contains an '_id' field with an '\$oid' value and an 'SGF' field with a long string of board game notation. The interface also includes a search bar, a 'Delete all' button, an 'Add document' button, and a 'Search' button.

圖表 37 MongoLab 搜尋結果 1

如圖表 37 MongoLab 搜尋結果，搜尋結果將會列在下方，一共是 421 筆棋形。

輕鬆分享圍棋棋譜

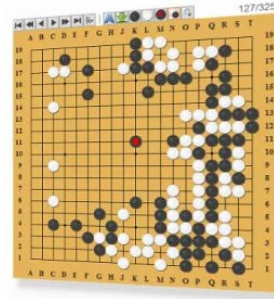
線上輸入棋譜

在各種瀏覽器中線上建立及分享棋譜，[開始打譜](#)。
其他選擇：[19路棋盤](#)、[13路棋盤](#)、[9路棋盤](#)。

上傳SGF檔案

未選擇任何檔案

貼上SGF



[圖文教學，詰棋製作輕鬆上手！](#)

Powered by © LGS

圖表 38 Lgs. tw 線上棋譜編輯器 1

如圖表 38 Lgs. tw 線上棋譜編輯器，為了能夠更清楚的了解搜尋結果，這裡使用了Lgs. tw 這個線上棋譜編輯器來呈現棋手順序，本網站只要貼上賽事SGF 訊息，即可呈現完整的下棋順序。

Documents

Query (blank returns all objects):

```
{
  "spboard": [
    [
      [0,0,0],
      [0,0,0],
      [0,2,2]
    ]
  ]
}
```

Sort order (1: asc, -1: desc):

Subset of fields (1: include, 0: exclude):

Search Results

Display mode: list table (edit table view)

records / page 10 [1 - 10 of 421] next > last >>

of];W[op];B[qm];W[oq];B[pr];W[pn];B[om];W[pm];B[pl];W[ol];B[nm];W[ok];B[qj];W[mo];B[mq];W[pj];B[ml];W[qi];B[rj];W[pg];B[pf];W[mj];B[qh];W[ln];B[kd];W[kj];B[lp];W[mf];B[rq];W[rp];B[rr];W[rm];B[rl];W[on];B[ni];W[ni];B[mk];W[mi];B[on];W[po];B[ro];W[ro];B[sm];W[km];B[li];W[il];B[im];W[lp];B[hm];W[an];B[gm];W[jm];B[jj];W[fm];W[dm];B[jg];W[cg];B[lc];W[mb];B[jc];[kf];B[jf];W[kc];B[ff];W[kg];B[jc];W[ib]

取消(U) Ctrl+Z
重做(R) Ctrl+Shift+Z
剪下(T) Ctrl+X
複製(C) Ctrl+C
貼上(P) Ctrl+V
以純文字貼上 Ctrl+Shift+V
刪除(D)

圖表 39 MongoLab 搜尋結果 2

輕鬆分享圍棋棋譜

線上輸入棋譜

在各種瀏覽器中線上建立及分享棋譜，[開始打譜](#)。

其他選擇：[19路棋盤](#)、[13路棋盤](#)、[9路棋盤](#)。

上傳SGF檔案

未選擇任何檔案



貼上SGF

```
(B[jp];W[pp];B[dc];W[dp];B[qd];W[ed];B[ec];W[nc];B[pc];  
;W[md];B[qf];W[ic];B[ce];W[fc];B[nq];W[qn];B[fg];W[cn];  
;B[pa];W[qa];B[qr];W[qe];B[or];W[dr];B[al];W[hq];B[hp];  
;W[ig];B[ip];W[ga];B[gp];W[fr];B[of];W[op];B[om];W[og];  
;B[er];W[er];B[em];W[em];B[el];W[ol];B[nm];W[ok];B[ql];
```

[圖文教學，詰棋製作輕鬆上手！](#)

Powered by © LGS

圖表 40 Lgs. tw 線上棋譜編輯器 2

如圖表 39 MongoLab 搜尋結果 2 &圖表 40 Lgs. tw 線上棋譜編輯器 2，將欲驗證欄位內的 SGF 複製到該網站，貼上至 Lgs. tw 的 SGF 輸入區域內，並按下送出。

請輸入棋譜的基本資訊，並檢查棋步

棋譜資訊：

標題：

執白：

執黑：

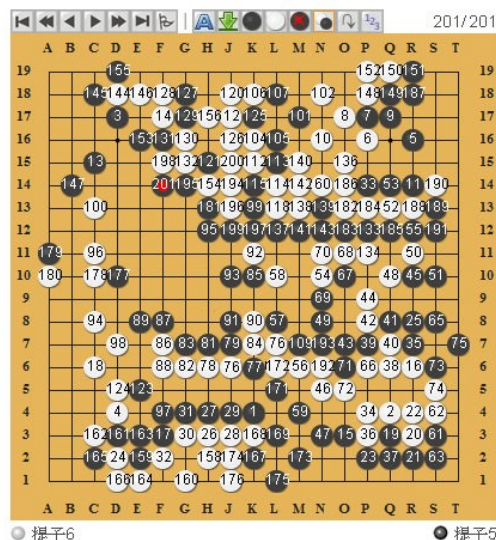
進階：

將棋譜連結寄送到我的信箱：

棋譜手數：**201**

檢查無誤後請點擊儲存鈕來保存棋譜：

棋盤：



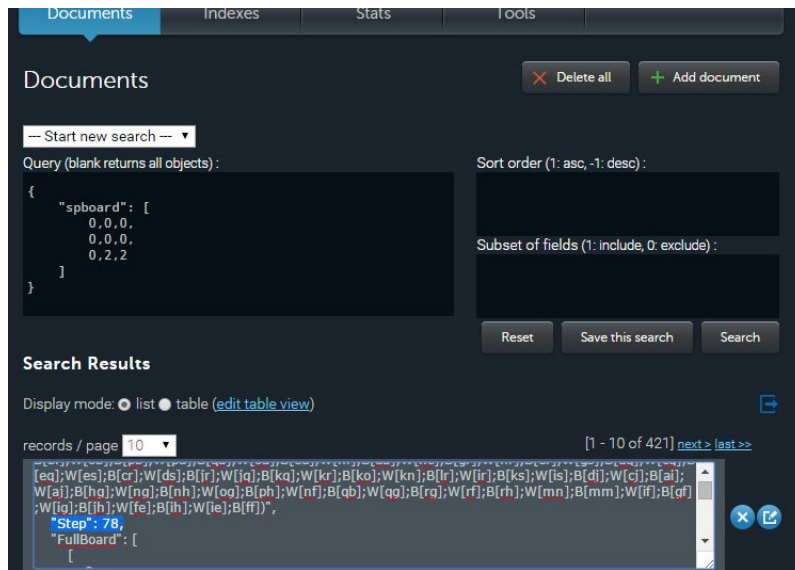
● 提子6

● 提子5

Powered by © LGS

圖表 41 Lgs. tw 線上棋譜編輯器 3

如圖表 41 Lgs. tw 線上棋譜編輯器 3，如此 Lgs. tw 便能將 SGF 轉換成黑白先後手順序供我們來驗證。



圖表 42 MongoLab 搜尋結果 3

如圖表 42 MongoLab 搜尋結果 3，回到 MongoLab 搜尋結果，方才搜尋到的棋形是位於第 78 手的順序。



圖表 43 Lgs. tw 線上棋譜編輯器 4

預設輸入完 SGF 會停留在最後一首，此範例為 201 手，在 Lgs. tw 調整至第 78 手的順序

請輸入棋譜的基本資訊，並檢查棋步

棋譜資訊:

標題:

執白:

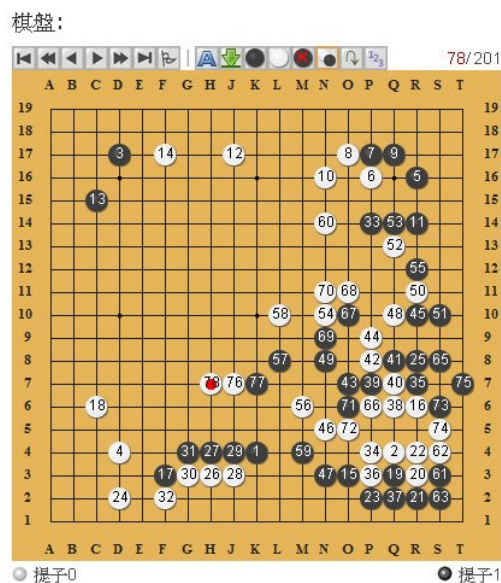
執黑:

進階:

將棋譜連結寄送到我的信箱:

棋譜手數: **78**

檢查無誤後請點擊儲存鈕來保存棋譜:



Powered by © LGS

圖表 44 Lgs. tw 線上棋譜編輯器 5

如圖表 44 Lgs. tw 線上棋譜編輯器 5，第 78 手時出現搜尋的棋形。



第五章、結論與展望

本論文設計並實作一個基於MongoDB的高效能棋譜搜尋系統。本系統包含了以下三個模組，棋譜格式的前置處理模組、資料轉換與儲存模組以及棋譜搜尋模組。

由於棋譜資料庫的資訊單純，本論文資料轉換與儲存模組採用 MongoDB 雲端資料庫作為圍棋資料庫，來取代坊間常用的關聯性資料庫的角色，發揮 MongoDB 於巨量資料處理上的優勢，借以提高系統效率。實驗成果使用 JAVA 為主要的程式設計出圖形化介面，讓使用者輸入欲尋的棋形，再讓系統從眾多賽事棋譜中找出相符的棋形，最後回傳該棋形的相關賽事資訊給使用者。

最後，目前MongoDB普遍還未被大多數的IT人員所接受，版本也時常處於不斷開發中的階段，升級後的不穩定以及相容性都還是一大課題



第六章、參考文獻

1. SGF 維基百科:
http://en.wikipedia.org/wiki/Smart_Game_Format
2. MongoDB 官方網站：<http://www.mongodb.org/>
3. 吳耀撰譯，2012，『MongoDB 技術手冊』，台北，碁峯資訊
4. 劉東岳，1989，「電腦圍棋程式之設計與製作」，國立臺灣大學資訊工程學研究所碩士論文
5. 徐國棟，2004，「圍棋棋譜棋形比對系統」，國立東華大學資訊工程所碩士論文
6. 朱仲康，2005，「圍棋棋譜之棋形資訊檢索」，國立東華大學資訊工程所碩士論文
7. 陳文誠，2008，利用 GPU 架構加速圍棋棋譜搜尋系統，長榮大學資訊管理研究所碩士論文
8. 黃珣，2012，MongoDB-植基於 YCSB 系統之叢集資料庫效能分析，國立中山大學資訊工程學系碩士論文
9. 劉東岳，1989，電腦圍棋程式之設計與製作，國立臺灣大學資訊工程學研究所碩士論文
10. M. Stonebraker, "SQL databases v. NoSQL databases," *Commun. ACM*, vol. 53, pp. 10-11, 2010.
11. Martin Mueller, "Computer Go", *Artificial Intelligence*, No.134, pp.145-179, 2002.
12. D.J.H.Brown and S.Dowsey, "The Challenge of GO", *New Scientist*, Pages 303-305.