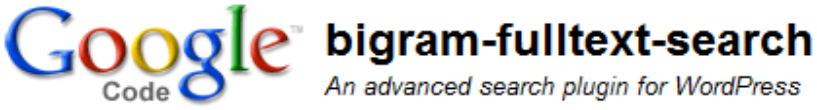
資料庫系統專題課程

期末報告

全文檢索搜尋

(Full Text Index Searching, FTIS)



指導老師：黃三益教授

D964020003 曾智義　Paul

D954020008 唐　筠　Tiffany

2009 年 01 月 10 日

目錄

圖目錄

摘要

網際網路的普及與資訊科技融入，締造網路無國界、資訊無所不在的環境，相對的也產生大量的知識、資訊、檔案與文件。web-based 的應用由 web2.0 概念的興起後，利用一個 web 平台，由使用者創造、協同合作、分享各種資訊與內容的分散式網路現象 [林東清, 2008]，而在以服務精神為導向與以使用者為中心的思維中，如何在大量的文件檔案中快速的找到需求的資訊與知識，如何提供完整且精確的檢索方式達到使用者需求，是一個值得探討的議題。

全文檢索的定義是資料中的全部文字以電子形式儲存，再以關鍵詞、索引典或自由詞彙等方式做為檢索的方法 [胡述兆, 1995]。本專案計畫擬由全文檢索基本概念和需求為基礎，簡單介紹與全文索引有關的議題，如：語幹處理 (stemming) 、符素解析器 (token parser)、斷詞 (word segmentation)、反向索引 (inverted index) ，還有基本的術語，如：召回率 (recall rate) 、精確率 (precision)等，彙整一些常用的全文檢索應用程式及全文檢索目前的應用，並以 SQL 2008 為技術與實作範例。本專案計畫的貢獻為由概念到技術實作與相關應用中，彙整全文檢索相關知識，提供一個完整性的介紹。

關鍵字：全文檢索、SQL 2008

1. 導論

隨著資訊科技的快速發展導致了各式各樣類型的資訊不斷的增加，例如：文字、圖形、語音，甚至多媒體型態等資訊。這些資訊大致可分爲兩大類：非結構化資料和結構化資料，結構化資料例如是：企業財務帳目和生產資料或是學生的分數資料等等；而非結構化資料，例如：文本資料或圖像聲音等多媒體資料等等。然而，以目前web 2.0等相關系統或應用的蓬勃發展，非結構化資料佔有整個信息量的絕大部份。對於結構化資料，傳統DBMS（資料庫管理系統，Database Management System）已能夠妥善管理資料的維護與查詢，但是對於非結構化資料若同樣採用結構化方式去管理則會顯得有些不當，特別是在查詢這些大量非結構化資料，在速度上不僅較慢也不易應付，因此後期新一代發展的DBMS全文檢索技術就能有效地管理與運用這些非結構化資料，並達到良好的查詢效能。

所謂全文檢索（Full Text Searching）簡單的說，是指：「在大量的文章裡找出某一段文字的方法」。從早期的循序搜尋（Sequential Searching），到現在的索引搜尋（Index Searching）都是為了能更快速更有效率的執行尋找比對的工作。而且在全文檢索系統中，首重索引結構（Index Structure），因為索引的結構除了影響索引建立的時間和索引檔的大小外更影響了查詢的速度。相較於全文檢索搜尋，LIKE Transact-SQL 語法只能針對字元模式運作，且也無法使用 LIKE 語法來查詢格式化的二進位資料。此外，針對大量非結構化文字資料執行 LIKE 查詢的速度會比針對相同資料執行對等全文檢索查詢的速度要慢很多。對於數百萬列的資料，使用 LIKE 查詢時可能要好幾分鐘才能傳回搜尋結果，但是使用全文檢索查詢時可能只要幾秒鐘的時間 (視傳回的資料列數目而定)。

全文檢索除了一般套用於應用程式中，目前也廣泛應用於各領域中，可以檢索文件、檔案、資料庫、網站等等不同來源，應用做為一個知識入口（Knowledge Protal , KP)，例如應用於電子商務、數位典藏、知識管理等相關領域，其功能涵

蓋網站目錄管理、搜尋系統、整合式搜尋以及自動分類、自動建構知識網路…等等。而全文檢索基本上是搜尋引擎的核心，當資料 (可能是文件、檔案或是網頁等) 被抓回硬碟中儲存之後，就可以利用全文檢索的機制進行檢索，其主要的技術有兩項 ：

1. 索引系統：建立檢索時使用的索引，使系統能在一兩次的硬碟存取後就查到資料。
2. 查詢系統：利用索引系統查詢資料後，將查到的資料呈現在顯示介面(例如：網頁) 上。

一般搜尋引擎的技術基礎是全文檢索，隨著Internet的發展，搜尋引擎在全文檢索技術上逐漸發展起來，並得到廣泛的應用，但搜尋引擎還是不同於全文檢索。搜尋引擎和全文檢索主要區別如下表：

表1 搜尋引擎和全文檢索主要區別 資料參考[中文全文檢索搜索網, 2004]

本文整理

	搜尋引擎	全文檢索
數據量	透過 Internet，網頁搜索需要處理幾十億的網頁，搜尋引擎的策略都是採用伺服器叢集和分散式運算技術	傳統全文檢索系統服務是企業本身的資料或者和企業相關的資料，一般索引庫規模多在 GB 級，資料量大的也只有幾百萬條
內容相關性	資訊太多，查詢準確率和排序就特別重要，Google 等搜尋引擎採用網頁連結分析技術，根據 Internet 上網頁被連結次數作為重要性評判的依據	全文檢索的資料來源中相互連結的程度並不高，不能作為判別重要性的依據，只能基於內容的相關性排序

安全性	Internet 上搜尋引擎的資料來源都是在其上的公開資訊，而且除了文本正文以外，其它資訊都不太重要	企業全文檢索的資料來源都是企業內部的資訊，有等級、許可權等限制，對查詢方式也有更嚴格的要求，因此其資料一般會安全和集中地存放在資料倉庫中以保證資料安全和管理的要求
個性化和智慧化	搜尋引擎面對的是 Internet 使用者，由於其資料量和客戶數量的限制，自然語言處理技術、知識檢索、知識挖掘等計算密集的智慧計算技術很難應用，這也是目前搜尋引擎技術努力的方向	全文檢索資料量小，檢索需求明確，客戶量少，在智慧化和個性化可有更長遠的發展

因此，全文檢索在未來的發展趨勢上與相關應用上仍有廣大的市場空間足以發揮，然而面對資訊科技日新月異的發展，為了更進一步了解整個全文檢索搜尋的機制與運作方式，本專題將對全文檢索做一初步的簡介其相關的概念，並以 SQL Server 2008 為基礎來介紹相關的技術與實作方式，且另外簡介其他有關全文檢索應用。

2. 全文檢索概念

全文檢索的方法主要分為依照「字」進行檢索和依照「詞」進行檢索兩種。依照「字」檢索是指對於文章中的每一個字都建立索引，檢索時將詞分解為字的組合。對於各種不同的語言而言，字有不同的含義，例如：在英文中，字與詞實際上是合一的，而在中文中，字與詞則有很大分別。依照「詞」檢索是指對文章中的詞，即語義單位建立索引，檢索時按詞檢索，並且可以處理同義項等。英文

等西方的文字，是依據空白進行切割分開「詞」，因此，在實行處理上與依照文書處理的模式是類似的，且添加同義處理也很容易，然而，中文等東方文字則需要進行切割分開字詞，以達到按詞索引的目的，目前中文的斷詞部分，已經有一些單位研發出正確率頗高的應用程式。

全文檢索系統是按照全文檢索理論建立起來的用於提供全文檢索服務的軟體系統。一般來說，全文檢索需要具備建立索引和提供查詢的基本功能，此外現代的全文檢索系統還需要具有方便的使用者介面、WWW 服務的開發介面、二次應用開發介面等等。功能上，全文檢索系統核心具有建立索引、處理查詢返回結果集、增加索引、優化索引結構等等功能，週邊則由各種不同應用具有的功能組成。結構上，全文檢索系統核心具有索引引擎、查詢引擎、文本分析引擎、對外介面等等，加上各種週邊應用系統等等共同構成了全文檢索系統。下圖為全文檢索系統架構示意圖：

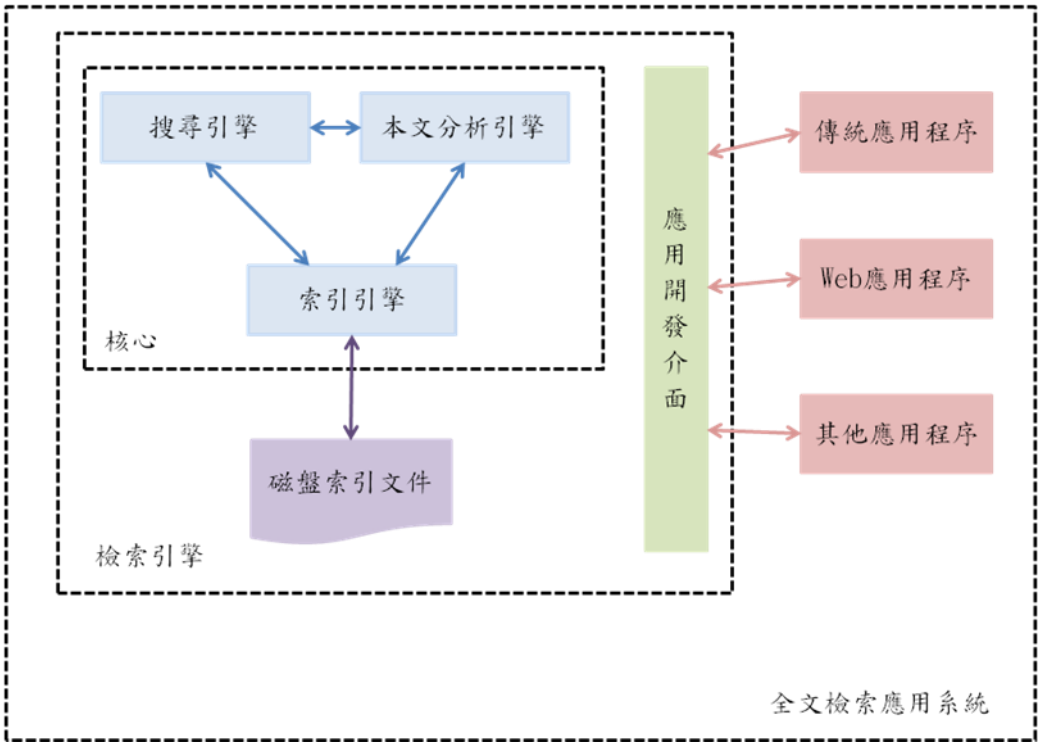


圖 1 全文檢索系統架構示意圖

2-1 語幹還原（stemming）

英文字彙中，動詞有原形、過去式、進行式、完成式的變化，名詞會有單複數的變化，而副詞有可能是由形容詞變化而來。所以為了達到字彙的精簡以利檢索，所以採用stemming的技術，stemming會把字彙還原成語幹的形式，如starting, starts, started都會被還原成start。

2-2 單元剖析（token parser）

所謂的 parser 是將符合既定文法(grammar)的文字轉換成結構化，進行全文檢索之前，將「詞」視為最小單位，即所謂的 token，然後透過一些相關規則語演算法等進行編譯，將 token 轉化為特定的資料結構。

2-3 斷詞（word segmentation）

對於英文而言，詞語詞之間是以空白做為間隔分開，而中文是以字為單位，句子中所有的字連起來才能描述一個意思。例如，英文句子「I am a student」，用中文敘述則為「我是一個學生」。電腦可以很簡單通過空格知道 student 是一個單詞，但是無法直接知道「學」、「生」兩個字合起來才表示「學生」一個詞。把中文的句子切分成有意義的詞，稱為中文分詞。

2-4 反向索引（inverted index）

反向索引是一種索引結構，它存儲了單詞與單詞它們自己本身在一個或多個文檔中所在位置之間的映射。反向索引通常利用關聯陣列實現，在全文檢索中，反向索引扮演重要角色。它擁有兩種表現形式：

1. *inverted file index*，其表現形式為｛單詞，單詞所在文檔的 ID｝
2. *full inverted index*，其表現形式為｛單詞，（單詞所在文檔的 ID，在

具體文檔中的位置）}

以實例說明，假設有三個文檔：

- T_0 = "it is what it is"
- T_1 = "what is it"
- T_2 = "it is a banana"

那麼，採用 inverted file index 方式，結果是：

"a": {2}

"banana": {2}

"is": {0, 1, 2}

"it": {0, 1, 2}

"what": {0, 1}

而採用 full inverted index 方式，結果是：

"a": {(2, 2)}

"banana": {(2, 3)}

"is": {(0, 1), (0, 4), **(1, 1)**, (2, 1)}

"it": {(0, 0), (0, 3), **(1, 2)**, (2, 0)}

"what": {(0, 2), **(1, 0)**}

2-5 召回率 (recall rate) 與精確度(precision)

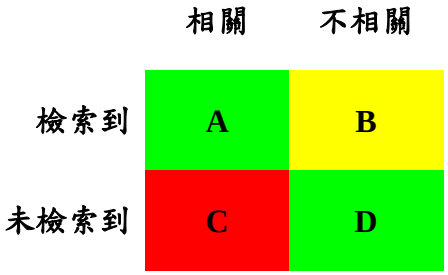
從一個大規模資料集合中檢索文件檔的時，可把文件檔分成下列四組，直觀的說，一個好的檢索系統檢索到的相關文件檔越多越好，不相關文件檔越少越好。召回率和精確度是衡量資訊檢索系統性能最重要的參數。

✧ 系統檢索到的相關文件檔（A）

✧ 系統檢索到的不相關文件檔（B）

✧ 相關但是系統沒有檢索到的文件檔（C）

✧ 相關但是被系統檢索到的文件檔（D）



✧ 召回率R：用檢索到相關文件檔數作為分子，所有相關文件檔總數作為分母，即 R=A/(A+C)

✧ 精確度P： 用檢索到相關文件檔數作為分子，所有檢索到的文件檔總數作為分母，即 P=A/(A+B)

下面舉例說明召回率和精確度之間的關係：一個資料庫有 500 個文件檔， 其中有 50 個文件檔符合定義的問題。系統檢索到 75 個文件檔,但是只有 45 個符合定義的問題。

✧ 召回率 R=45/50=90%

✧ 精確度 P=45/75=60%

本例中，系統檢索是比較有效的，召回率為 90%。但是結果對一位使用者而言，並非其所需求的，因為有近一半的檢索結果是不相關。研究表明：在不犧牲精確度的情況下，獲得一個高召回率是很困難的。召回率越高，精確度下降的很快，而且這種趨勢不是線性的。

3. 資料庫設計與全文檢索

對於處理大量資料而言，資料庫設計大概分為三類 [徐玉梅, 2002]：

a. 以”文檔”結構搭配不同的索引方法：如逐項反轉索引法（Inversion of Terms）、簽名檔（ Signature File ）、聚叢法（Clustering ）、字典式索引法（ Dictionary-Based Indexing ）或字典推論規則式索引法（Dictionary/Rule-based Indexing）等所建立 的資料庫系統。

b. 關聯式資料庫管理系統(RDBMS)：如IBM DB2、Oracle、Verity、Sybase、Informix 及微軟SQL Server, ACCESS 等，通常是將關聯式資料庫中某些欄位改變成變動長度的全文式欄位，在這些欄位下提供全文檢索功能。

c. 原生性XML資料庫：資料儲存是採”文檔”格式，按照階層化結構存放，而描述其組成元素的屬性是在DTD 中予以聲明，雖然它也是”文檔”格式，但因係新的發展且具有相當的獨特性，故將獨立為一類

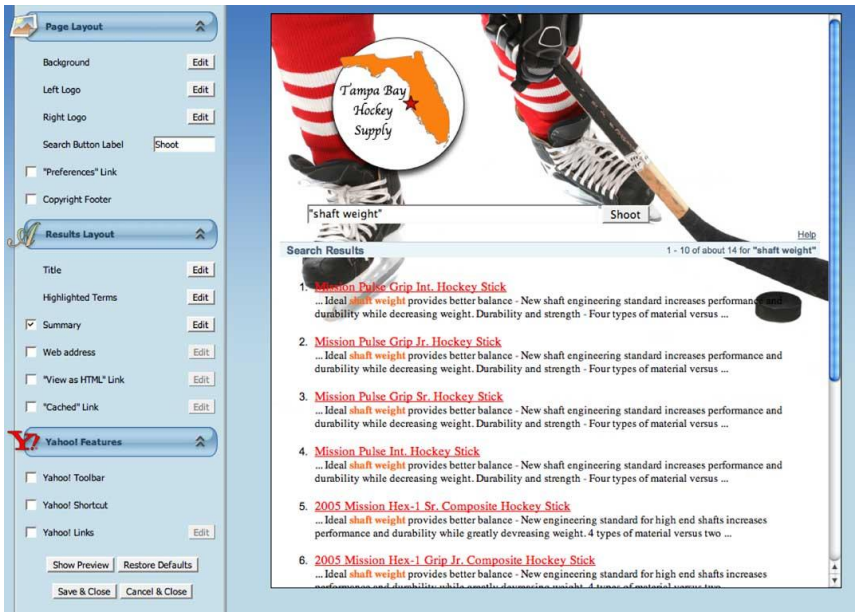
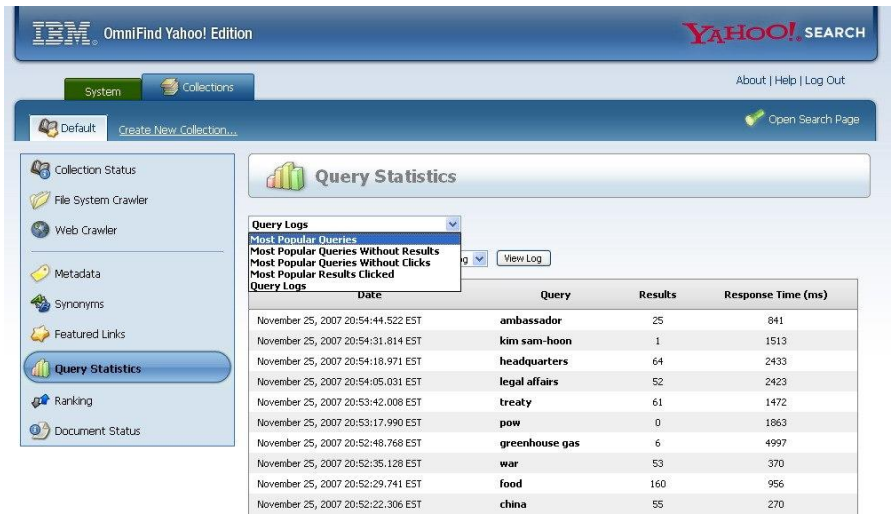
4. 常見全文檢索應用系統

目前市面上常見有關全文檢索相關應用程式非常多，一般企業界應用於文件檔案管理與知識資產管理，或是個人的文件檔案管理，或是專利文件、數位典藏系統等。下列以全文檢索引擎與套裝應用程式、提供全文檢索之資料庫管理系統作為不同分類，分別說明常見的全文檢索應用系統如下：

4-1 全文檢索引擎與套裝應用程式

4-1-1 IBM OmniFind Yahoo! Edition

使用平台	跨平台支援 (32 位元的 Linux 與 Windows 都支援)
搜尋引擎	OmniFind Yahoo! Edition 是 IBM 與 Yahoo! 公司合作開發的搜尋軟體
語言	多國語言
介面	API 介面，搜尋功能整合既有系統
索引建立	有下列三種方式：

	1. 直接針對檔案系統建立索引 2. 透過 Web Crawler (Spider) 直接對特定網址進行全站檢索 3. 可透過 API 進行文件索引
同義字	支援**同義字**，並可匯入、匯出
官方網址	http://www.demos.ibm.com/on_demand/Demo/zh_tw/IBM_Demo_IBM_OmniFind_Yahoo_Edition-Dec06.html
客製化搜尋介面	 內建客製化搜尋介面的設定，也提供 API 介面讓你客製化搜尋結果
統計功能	 查詢狀態統計功能

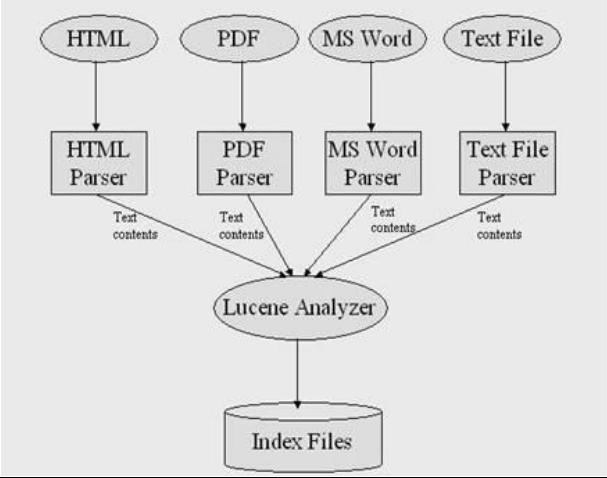
費用	有免費版可以下載，也有付費版

範例	IBM Omnifind Yahoo! Edition and C# 建立搜尋索引參考範例
	➢ Searching(建立搜尋索引)
	1. 建立 XML 與索引文字庫到 aspx 網頁
	<asp:TextBox runat="server" ID="txtSearch" />
	
	<asp:Xml runat="server" TransformSource="~/search.xsl" EnableViewState="false" ID="xmlSearch" />
	
	2. 建立查詢字串，強化搜尋能力
	string strSearch = txtSearch.Text;
	string strRequest = "http://[omnifindserver:port#]/api/search?query=" + strSearch + "&collection=Default";
	
	3. 將找到和讀到的結果回傳
	string strResults = "";
	strResults = new System.Net.WebClient().DownloadString(strRequest);
	xmlSearch.DocumentContent = strResults;

相關網址	1. Installing OmniFind Yahoo! Edition Version 8.4.2 2. Administration Guide 3. Developer Guide

4-1-2 Lucene.NET

與 Lucene	1. 2002/7/7 ： NLucene 1.2 在 SourceForge 網站上註冊，將 Lucene

之淵源	1.2 改用 C#撰寫，開始了 Lucene 在 Microsoft .NET 平台上的 porting version。然而在 NLucene 1.2 beta 2 之後，NLucene 就停頓了下來。 2. 2003 年：另一個.NET 平台的 Lucene porting version - LuceneDotNet 在 SourceForge 網站上頭註冊，繼續了原本 NLucene 停頓的開發工作，將 Lucene 1.3 以 C#撰寫，以供.NET 程式設計師使用。 3. 2004 年 11 月：LuceneDotNet 改名叫 dotLucene，但 dotLucene 的開發團隊在 2006 年 7 月之後，因為無力繼續維護而轉交給 Apache Software Foundation 維護，並改名叫做 Lucene.NET。
官方網站	http://incubator.apache.org/lucene.net/
性質	是一個全文索引引擎工具包
索引架構	1. Lucene 使用各種解析器對各種不同類型的文檔進行解析，例如對於 HTML 文檔。HTML 解析器會做一些預處理的工作，例如：過濾文檔中的 HTML 標籤等等。HTML 解析器的輸出的是文本內容，接著 Lucene 的分詞器(Analyzer)從文本內容中提取出索引項目以及相關資訊，例如索引項目的出現頻率。接著 Lucene 的分詞器把這些資訊寫到索引檔中。 2. Lucene 的索引機制的架構 

特點	利用 Lucene，在建置索引的工程中可以充分利用機器的硬體資源來提高索引的效率。如果需要索引大量的檔時，會出現索引過程的瓶頸是在往磁片上寫索引檔的過程中。為了解決這個問題, Lucene 在記憶體中持有一塊緩衝區。但如何控制 Lucene 的緩衝區呢？可以使用 Lucene 的 IndexWriter 提供了三個參數用來調整緩衝區的大小以及往磁片上寫索引檔的頻率。 1．合併因數 (mergeFactor) 這個參數決定了在 Lucene 的一個索引塊中可以存放多少文檔，以及把磁片上的索引塊合併成一個大的索引塊的頻率。例如：如果合併因數的值是 10，那麼當記憶體中的文檔數達到 10 的時候所有的文檔都必須寫到磁片上的一個新的索引塊中。並且，如果磁片上的索引塊的隔數達到 10 的話，這 10 個索引塊會被合併成一個新的索引塊。這個參數的預設值是 10，如果需要索引的文檔數非常多的話這個值將是非常不合適的。對批次處理的索引來講，為這個參數賦一個比較大的值會得到比較好的索引效果。 2．最小合併文檔數 (minMergeDocs) 這個參數也會影響索引的性能。它決定了記憶體中的文檔數至少達到多少才能將它們寫回磁片。這個參數的預設值是 10，如果有足夠的記憶體，那麼將這個值儘量設的比較大一些將會顯著的提高索引性能。 3．最大合併文檔數 (maxMergeDocs) 這個參數決定了一個索引塊中的最大的文檔數。它的預設值是

	Integer.MAX_VALUE，將這個參數設置為比較大的值可以提高索引效率和檢索速度，由於該參數的預設值是整型的最大值，所以一般不需要改動這個參數。
相關資源	Lucene 相關資源： 1. Lucene 官方網站 2. Apache Lucene 3. Lucene FAQ 4. Lucene.Net 5. Lucene API (Java) 6. DotLucene 7. Luke - Lucene Index Toolbox 8. Nutch 9. LUCENE.COM.CN 中國 10. Compass 11. 實戰 Lucene，第 1 部分: 初識 Lucene 12. 深入 Lucene 索引機制
範例： 基本應用	using System; using System.Collections.Generic; using System.Text; using Lucene.Net; using Lucene.Net.Analysis; using Lucene.Net.Analysis.Standard; using Lucene.Net.Documents; using Lucene.Net.Index; using Lucene.Net.QueryParsers;

```
using Lucene.Net.Search;
using Lucene.Net.Store;
using Lucene.Net.Util;
/* 應用控制平台
namespace ConsoleApplication1.Lucene
{
   public class LuceneTest
   {
      private const string FieldName = "name";
      private const string FieldValue = "value";
      private Directory directory = new RAMDirectory();
      private Analyzer analyzer = new StandardAnalyzer();
      public LuceneTest()
      {
      }
      private void Index()       /* 建置索引
      {
         IndexWriter writer = new IndexWriter(directory, analyzer,
true);
         writer.maxFieldLength = 1000;
          for (int i = 1; i <= 100; i++)
         {
            Document document = new Document();
            document.Add(new Field(FieldName, "name" + i,
Field.Store.YES, Field.Index.UN_TOKENIZED)); /*拆解字元
```

```
            document.Add(new Field(FieldValue, "Hello, World!",
Field.Store.YES, Field.Index.TOKENIZED));
            writer.AddDocument(document);
        }
        writer.Optimize();   /* 最佳化
        writer.Close();
    }
    private void Search()   /*搜尋
    {
        Query query = QueryParser.Parse("name*", FieldName,
analyzer);
        IndexSearcher searcher = new IndexSearcher(directory);
        Hits hits = searcher.Search(query);
        Console.WriteLine("符合條件記錄:{0}; 索引庫記錄總
數:{1}", hits.Length(), searcher.Reader.NumDocs());

        for (int i = 0; i < hits.Length(); i++)
        {
            int docId = hits.Id(i);
            string name = hits.Doc(i).Get(FieldName);
            string value = hits.Doc(i).Get(FieldValue);
            float score = hits.Score(i);
            Console.WriteLine("{0}: DocId:{1}; Name:{2}; Value:{3};
Score:{4}",
                i + 1, docId, name, value, score);
```

	}
	searcher.Close();
	}
	}
	}
	
	除了 RAMDirectory，還可以使用 FSDirectory
	(注意 FSDirectory.GetDirectory 的 create 參數，為 true 時將刪除已有索引庫檔，可以通過 IndexReader.IndexExists() 方法判斷。)
	
	/* 從指定目錄打開已有索引庫。
	private Directory directory = FSDirectory.GetDirectory("c:\index", false);
	
	/* 將索引庫載入記憶體，以提高搜索速度
	private Directory directory = new RAMDirectory(FSDirectory.GetDirectory(@"c:\index", false));
	//或
	//private Directory directory = new RAMDirectory(c:\index");

範例： 多欄位搜尋	/* 使用 MultiFieldQueryParser 可以指定多個搜索欄位 /* 剖析查詢字串 Query query = MultiFieldQueryParser.Parse("name*", new string[] { FieldName, FieldValue }, analyzer); IndexReader reader = IndexReader.Open(directory); IndexSearcher searcher = new IndexSearcher(reader); Hits hits = searcher.Search(query);

範例：
多條件搜尋
	/* 除了使用 QueryParser.Parse 分解複雜的搜索語法外，還可以通過組合多個 Query 來達到目的
	// 詞語搜索
	Query query1 = new TermQuery(new Term(FieldValue, "name1"));
	
	// 萬用字元
	Query query2 = new WildcardQuery(new Term(FieldName,
	"name*"));
	
	/ 欄位搜索 Field:Keyword，自動在結尾添加 *
	//Query query3 = new PrefixQuery(new Term(FieldName, "name1")); /
	//Query query4 = new RangeQuery(new Term(FieldNumber,
	NumberTools.LongToString(11L)), new Term(FieldNumber,
	
	// 範圍搜索
	NumberTools.LongToString(13L)), true);
	
	/ *帶過濾條件的搜索
	//Query query5 = new FilteredQuery(query, filter); /
	
	BooleanQuery query = new BooleanQuery();
	query.Add(query1, BooleanClause.Occur.MUST);
	query.Add(query2, BooleanClause.Occur.MUST);
	
	IndexSearcher searcher = new IndexSearcher(reader);

	Hits hits = searcher.Search(query);

範例：
設置權重
	可以給 Document 和 Field 增加權重(Boost)，使其在搜索結果排名更加靠前。初始情況下，搜索結果以 Document.Score 作為排序依據，該數值越大排名越靠前。Boost 初始值為 1。
	
	Score = Score * Boost
	通過上面的公式，我們就可以設置不同的權重來影響排名。
	
	如下面的例子中根據 VIP 級別設定不同的權重。
	Document document = new Document();
	switch (vip)
	{
	case VIP.Gold: document.SetBoost(2F); break;
	case VIP.Argentine: document.SetBoost(1.5F); break;
	}
	只要 Boost 足夠大，那麼就可以讓某個命中結果永遠排第一位

範例： 排序	/ * 通過 SortField 的構造參數，我們可以設置排序欄位，排序條件，以及反向 Sort sort = new Sort(new SortField(FieldName, SortField.DOC, false)); IndexSearcher searcher = new IndexSearcher(reader); Hits hits = searcher.Search(query, sort); 排序對搜索速度影響還是很大的，盡可能不要使用多個排序條件
範例：	/* 使用 Filter 對搜索結果進行過濾，可以獲得更小範圍內更精確

過濾	的結果。
	
	舉個例子，我們搜索上架時間在 2005-10-1 到 2005-10-30 之間的商品，對於日期時間，我們需要轉換一下才能添加到索引庫，同時還必須是索引欄位。
	
	// index
	document.Add(FieldDate, DateField.DateToString(date),
	Field.Store.YES, Field.Index.UN_TOKENIZED);
	
	//...
	
	// search
	Filter filter = new DateFilter(FieldDate, DateTime.Parse("2005-10-1"),
	DateTime.Parse("2005-10-30"));
	Hits hits = searcher.Search(query, filter);
	
	除了日期時間，還可以使用整數。例如：搜索價格在 100 ~ 200 之間的商品。Lucene.Net NumberTools 對於數位進行了補位元處理，如果需要使用浮點數可以自己參考源碼進行
	
	// index
	document.Add(new Field(FieldNumber,
	NumberTools.LongToString((long)price), Field.Store.YES,
	Field.Index.UN_TOKENIZED));

```
//...


// search
Filter filter = new RangeFilter(FieldNumber,
NumberTools.LongToString(100L),
NumberTools.LongToString(200L), true, true);
Hits hits = searcher.Search(query, filter);


/ * 使用 Query 作為過濾條件
QueryFilter filter = new QueryFilter(QueryParser.Parse("name2",
FieldValue, analyzer));


/* 使用 FilteredQuery 進行多條件過濾
Filter filter = new DateFilter(FieldDate,
DateTime.Parse("2005-10-10"), DateTime.Parse("2005-10-15"));
Filter filter2 = new RangeFilter(FieldNumber,
NumberTools.LongToString(11L), NumberTools.LongToString(13L),
true, true);


Query query = QueryParser.Parse("name*", FieldName, analyzer);
query = new FilteredQuery(query, filter);
query = new FilteredQuery(query, filter2);


IndexSearcher searcher = new IndexSearcher(reader);
Hits hits = searcher.Search(query);
```

範例： 分佈搜索	/* 使用 MultiReader 或 MultiSearcher 搜索多個索引庫 MultiReader reader = new MultiReader(new IndexReader[] { IndexReader.Open(@"c:\index"), IndexReader.Open(@"\\server\index") }); IndexSearcher searcher = new IndexSearcher(reader); Hits hits = searcher.Search(query); 或 IndexSearcher searcher1 = new IndexSearcher(reader1); IndexSearcher searcher2 = new IndexSearcher(reader2); MultiSearcher searcher = new MultiSearcher(new Searchable[] { searcher1, searcher2 }); Hits hits = searcher.Search(query); 還可以使用 ParallelMultiSearcher 進行多執行緒並行搜索

範例：
合併索引庫
	/* 將 directory1 合併到 directory2 中
	Directory directory1 = FSDirectory.GetDirectory("index1", false);
	Directory directory2 = FSDirectory.GetDirectory("index2", false);
	
	IndexWriter writer = new IndexWriter(directory2, analyzer, false);
	writer.AddIndexes(new Directory[] { directory });
	Console.WriteLine(writer.DocCount());
	writer.Close();

範例： 操作索引庫	/* 刪除 (軟刪除，僅添加了刪除標記。調用 IndexWriter.Optimize() 後真正刪除)

	IndexReader reader = IndexReader.Open(directory);
	
	// 刪除指定序號(DocId)的 Document。
	reader.Delete(123);
	
	// 刪除包含指定 Term 的 Document。
	reader.Delete(new Term(FieldValue, "Hello"));
	
	// 恢復軟刪除
	reader.UndeleteAll();
	
	reader.Close();
	
	/* 增量更新
	(只需將 create 參數設為 false，即可往現有索引庫添加新資料)
	
	Directory directory = FSDirectory.GetDirectory("index", false);
	IndexWriter writer = new IndexWriter(directory, analyzer, false);
	writer.AddDocument(doc1);
	writer.AddDocument(doc2);
	writer.Optimize();
	writer.Close();

優化	/* 批量向 FSDirectory 增加索引時，增大合併因數(mergeFactor)和最小文檔合併數(minMergeDocs)有助於提高性能，減少索引時間

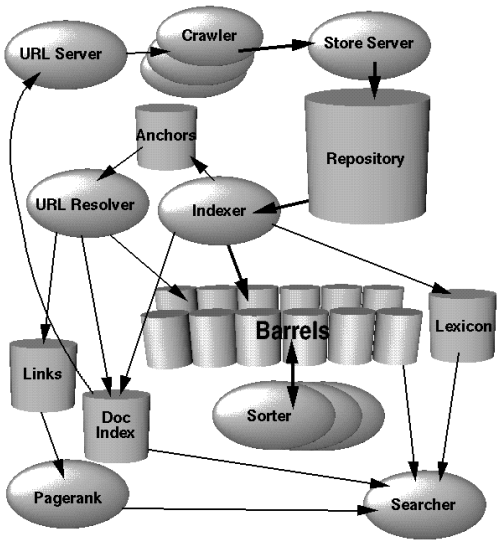
	IndexWriter writer = new IndexWriter(directory, analyzer, true);
	writer.maxFieldLength = 1000; // 欄位最大長度
	writer.mergeFactor = 1000;
	writer.minMergeDocs = 1000;
	
	for (int i = 0; i < 10000; i++)
	{
	// Add Documentes...
	}
	
	writer.Optimize();
	writer.Close();

範例： 不記錄檔案名稱的索引	/* 透過 document 物件來為特定檔案建立索引，但是並不儲存檔案名稱（亦即不把檔案名稱列入 field 中） Document doc = new Document(); doc.Add(new Field(”filename”, file.FullName, Field.Store.NO, Field.Index.UN_TOKENIZED)); doc.Add(new Field(”contents”, (new StreamReader(file.FullName)).ReadToEnd(), Field.Store.YES, Field.Index.TOKENIZED)); 如此仍可以正確將檔案建立索引，但在搜尋時只可以得到有幾筆吻合的資料，但是將無法列出檔案名稱

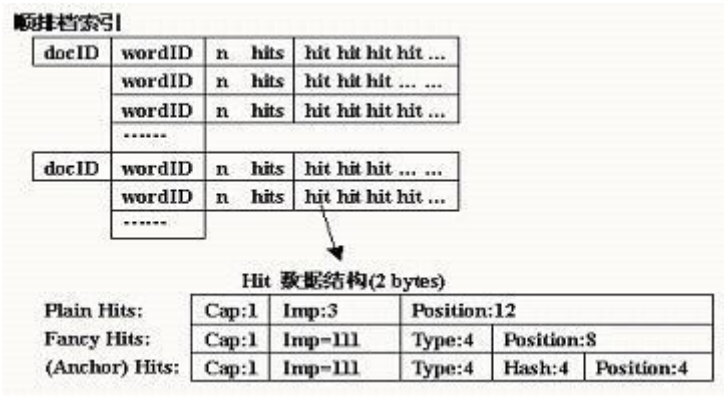
範例： 將檔案路徑一併加入索引	/* 透過 document 物件來為特定檔案建立索引，但是在索引的部份一並將檔案的路徑加入索引當中 Document doc = new Document(); doc.Add(new Field(”filename”, file.FullName, Field.Store.YES, Field.Index.UN_TOKENIZED)); doc.Add(new Field(”contents”, (new StreamReader(file.FullName)).ReadToEnd() + ” ” + file.FullName, Field.Store.YES, Field.Index.TOKENIZED)); 如此不僅可以正確將檔案內容建立索引，且在搜尋時還可以輸入檔案路徑的部份關鍵字來作搜尋，我們輸入 christmas 仍可以找到結果
範例： 一般的索引	doc.Add(new Field(”filename”, file.FullName, Field.Store.YES, Field.Index.UN_TOKENIZED)); doc.Add(new Field(”contents”, (new StreamReader(file.FullName)).ReadToEnd(), Field.Store.YES, Field.Index.TOKENIZED));

4-1-3 Google 搜尋引擎

特點	Google 搜尋引擎有兩個重要特點，有助於得到高精度的搜索結果 1. 應用 Web 的連結結構計算每個網頁的 Rank 值，稱為 PageRank 2. Google 利用超連結改進搜索結果
演算法	主要以 PageRank 演算法為主，此演算法對於 gogle 所提出的全文檢索有很大的助益

工作流程	 Sergey Brin and Lawrence Page, The Anatomy of a Large-Scale Hypertextual. Web Search Engine, 1998
說明	1. Google 使用高速的分散式爬行器(Crawler)系統中的漫遊遍歷器(Googlebot)定時地遍歷網頁，將遍歷到的網頁送到存儲伺服器(Store Server)中。 2. 存儲伺服器使用 zlib 格式壓縮軟體將這些網頁進行無失真壓縮處理後存入資料庫 Repository 中。Repository 獲得了每個網頁的完全 Html 代碼後，對其壓縮後的網頁及 URL 進行分析，記錄下網頁長度、URL、URL 長度和網頁內容，並賦予每個網頁一個文檔號(docID)，以便當系統出現故障的時候，可以及時完整地進行網頁的資料恢復。 3. 索引子(Indexer)從 Repository 中讀取資料，以後做以下四步工作： 4. 將讀取的資料解壓縮後進行分析，它將網頁中每個有意義的詞進行統計後，轉化為關鍵字(wordID)的若干索引項目(Hits)，生成索引項目清單，該清單包括關鍵字、關鍵字的位置、關鍵字的大小和大小寫狀態等。索引項目清單被存入到資料桶

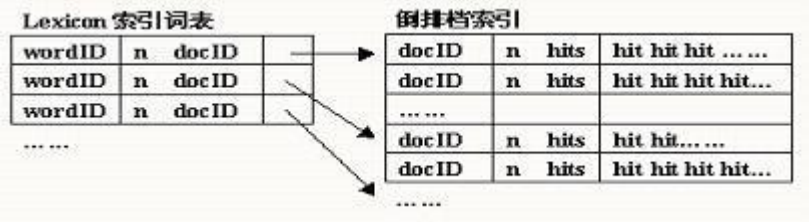
(Barrels)中，並生成以文檔號(docID)部分排序的順排檔索引。順排檔索引和 Hit 的存儲結構如圖：



5. 索引項目根據其重要程度分為兩種：當索引項目中的關鍵字出現在 URL、標題、錨文本(Anchor Text)和標籤中時，表示該索引項目比較重要，稱為特殊索引項目(Fancy Hits)；其餘情況則稱為普通索引項目(Plain Hits)。在系統中每個 Hit 用兩個位元組(byte)存儲結構表示：特殊索引項目用 1 位(bit)表示大小寫，用二進位碼 111(占 3 位)表示是特殊索引項目，其餘 12 位有 4 位表示特殊索引項目的類型(即 hit 是出現在 URL、標題、連結結點還是標籤中)，剩下 8 位表示 hit 在網頁中的具體位置；普通索引項目是用 1 位表示大小寫，3 位元表示字體大小，其餘 12 位元表示在網頁中的具體位置。

值得注意的是，當特殊索引項目來自 Anchor Text 時，特殊索引項目用來表示位置的資訊（8 位元）將分為兩部分：4 位表示 Anchor Text 出現的具體位置，另 4 位則用來與表示 Anchor Text 所連結網頁的 docID 相連接，這個 docID 是由 URL Resolver 經過轉化存入順排檔索引的。

	6. 索引子除了對網頁中有意義的詞進行分析外，還分析網頁的所有超文字連結，將其 Anchor Text、URL 指向等關鍵資訊存入到 Anchor 文件庫中。 7. 索引子生成一個索引詞表(Lexicon)，它包括兩個部分：關鍵字的清單和指標清單，用於倒排檔文檔相連接。 8. 索引子還將分析過的網頁編排成一個與 Repository 相連接的文檔索引(Document Index)，並記錄下網頁的 URL 和標題，以便可以準確查找出在 Repository 中存儲的原網頁內容。而且把沒有分析的網頁傳給 URL Server，以便在下一次工作流程中進行索引分析。 9. URL 分析器（URL Resolver）讀取 Anchor 文檔中的資訊，然後做第 10 點中的工作。 10. (a) 將其錨文本(Anchor Text)所指向的 URL 轉換成網頁的 docID；(b)將該 docID 與原網頁的 docID 形成“連結對”，存入 Link 資料庫中；(c)將 Anchor Text 指向的網頁的 docID 與順排檔特殊索引項目 Anchor Hits 相連接。 11. 資料庫 Link 記錄了網頁的連結關係，用來計算網頁的 PageRank 值。 12. 文檔索引(Document Index)把沒有進行索引分析的網頁傳遞給 URL Server，URL Server 則向 Crawler 提供待遍歷的 URL，這樣，這些未被索引的網頁在下一次工作流程中將被索引分析。 13. 排序器（Sorter）對資料桶(Barrels)的順排檔索引重新進行排序，生成以關鍵字(wordID)為索引的倒排檔索引。 14. 倒排檔索引結構如圖

	 Google 倒排檔索引結構 15. 將生成的倒排檔索引與先前由索引子產生的索引詞表(Lexicon)相連接產生一個新的索引詞表供搜索器(Searcher)使用。搜索器的功能是由網頁伺服器實現的，根據新產生的索引詞表結合上述的文檔索引(Document Index)和 Link 資料庫計算的網頁 PageRank 值來匹配檢索。 16. 在執行檢索時，Google 通常遵循以下步驟（以下所指的是單個檢索詞的情況）： (1)將檢索詞轉化成相應的 wordID； (2)利用 Lexicon，檢索出包含該 wordID 的網頁的 docID； (3)根據與 Lexicon 相連的倒排檔索引，分析各網頁中的相關索引項目的情況，計算各網頁和檢索詞的匹配程度，必要時調用順排檔索引； (4)根據各網頁的匹配程度，結合根據 Link 產生的相應網頁的 PageRank 情況，對檢索結果進行排序； (5)調用 Document Index 中的 docID 及其相應的 URL，將排序結果生成檢索結果的最終列表，提供給檢索用戶。 17. 使用者檢索包含多個檢索詞的情況與以上單個檢索詞的情況類似：先做單個檢索詞的檢索，然後根據檢索式中檢索符號的要求進行必要的布林操作或其他操作。
官方網址	http://www.google.com.tw/

4-2 提供全文檢索之資料庫管理系統

4-2-1 Oracle

緣由	1. Oracle 從 7.3 開始支援全文檢索，即用戶可以使用 Oracle 伺服器的上下文（ConText）選項完成基於文本的查詢。具體可以採用萬用字元查找、模糊匹配、相關分類、近似查找、條件加權和詞意擴充等方法。 2. 在 Oracle8.0.x 中稱為 ConText ；在 Oracle8i 中稱為 interMedia Text ； Oracle9i 中稱為 Oracle Text。 3. Oracle Text 是 9i 標準版和企業版的一部分。Oracle9i 將全文檢索功能做為內置功能提供給用戶，使得使用者在創建資料庫實例時自動安裝全文檢索。 4. Oracle Text 的應用領域有很多： a. 搜索文本 ：需要快捷有效搜索文本資料的應用程式 b. 管理多種文檔：允許搜索各種混和文檔格式的應用程式，包括 ord, excel, lotus 等 c. 從多種資料來源中檢索文本：不僅來自 Oracle 資料庫中的文本資料，而且可以來自 Internet 和檔案系統的文本資料 d. 搜索 XML 應用程式
搜索文件	1. 不使用 Oracle text 功能，也有很多方法可以在 Oracle 資料庫中搜索文本。可以使用標準的 INSTR 函數和 LIKE 操作符實現 SELECT * FROM mytext

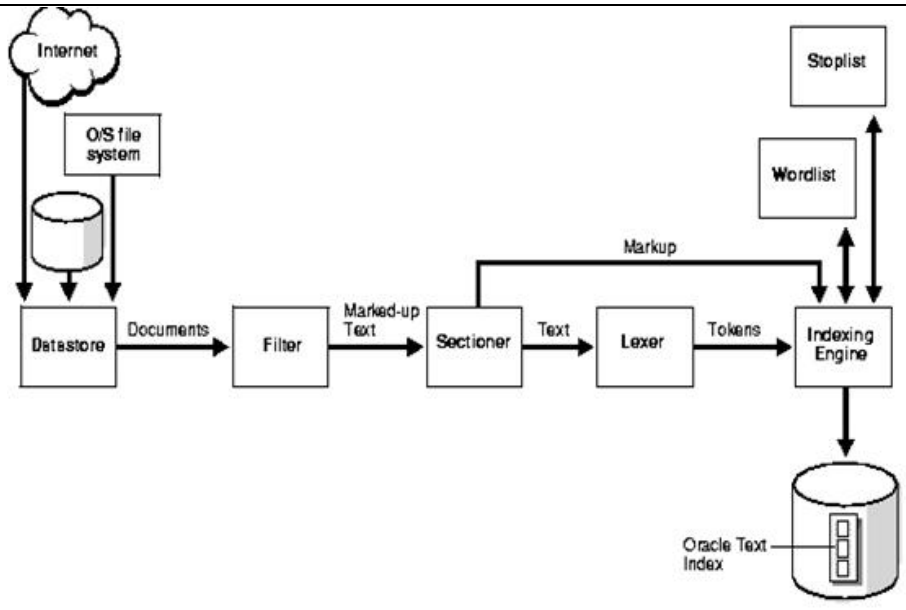
WHERE INSTR (thetext, 'Oracle') > 0;

```
SELECT *
  FROM mytext
  WHERE thetext LIKE '%Oracle%';
```

2. 搜索僅跨越很小的表的時候，使用 instr 和 like 是很理想的方法。然而通過這些文本定位的方法將導致全資料表掃描，對資源來說消耗比較昂貴，而且實現的搜索功能也非常有限.
3. 利用 Oracle Text，你可以回答如“在存在單詞‘Oracle’的行同時存在單詞’Corporation’，而且兩單詞間距不超過 10 個單詞的文本‘,’查詢含有單詞’Oracle’或者單詞’california’的文本，並且將結果按準確度進行排序‘,’含有詞根 train 的文本。以下的 sql 程式碼實現了如上功能

```
DROP   INDEX index mytext_idx
/
CREATE INDEX mytext_idx
ON mytext( thetext )
INDEXTYPE is CTXSYS.CONTEXT
/
SELECT id
  FROM mytext
  WHERE contains (thetext, 'near((Oracle,Corporation),10)') > 0
```

	／ SELECT score (1), id FROM mytext WHERE contains (thetext, 'Oracle or california', 1) > 0 ORDER BY score (1) DESC / SELECT id FROM mytext WHERE contains (thetext, '$train') > 0;
環境建置	1. 首先檢查資料庫中是否有 **CTXSYS** 使用者和 **CTXAPP** 腳色。如果沒有這個使用者和角色，意味著你的資料庫建置時未安裝 intermedia 功能 2. Oracle 一般是通過所謂的‘外部調用功能’(external procedure) 來實現 intermedia 的。 3. 語法：Create a user/table/index/query thus: As SYS or SYSTEM ---------- CREATE USER ctxtest IDENTIFIED BY ctxtest; GRANT CONNECT, RESOURCE, ctxapp TO ctxtest; ---------- Do any other grants, quotas, tablespace etc. for the new user. As CTXTEST:

	---------- CREATE TABLE quick (quick_id NUMBER PRIMARY KEY, text VARCHAR(80)); INSERT INTO quick (quick_id, text) VALUES (1, 'The cat sat on the mat'); INSERT INTO quick (quick_id, text) VALUES (2, 'The quick brown fox jumped over the lazy dog'); COMMIT ; CREATE INDEX quick_text ON quick (text) INDEXTYPE IS ctxsys.CONTEXT; ----------
運作機制	

索引概念	1. 利用 Oracle Text 對文檔集合進行檢索的時候，你必須先在你的文本列上建立索引。索引將文本打碎分成很多記號（token），這些記號通常是用空格分開的一個個單詞。 2. Oracle Text 應用的實現實際上就是一個 資料裝載—> 索引資料—>執行檢索 的一個過程 3. 建立的 Oracle Text 索引被稱為域索引（domain index），包括 4 種索引類型，利用 Create Index 建立這 4 種索引： a. CONTEXT b. CTXCAT c. CTXRULE d. CTXXPATH 4. CONTEXT 索引是以 CONTAINS 為查詢操作符號，其描述如下： 用於對含有大量連續文本資料進行檢索。支援 word、html、xml、text 等很多資料格式。支援中文字元集，支援分區索引，唯一支援並行創建索引（Parallel indexing）的索引類型。 對表進行 DML 操作後，並不會自動同步索引。需要手工同步索引 5. CONTEXT 索引的結構 Oracle Text CONTEXT 索引是反向索引（inverted index）。每個記號 （token）都映射著包含它自己的文本位置。 6. 例如 ‘I Love www.itpub.net’將會被分成 I ,LOVE,WWW,ITPUB,NET 這樣的記號（token）。 在索引建立過程中，單詞 Cat 會包括如下：

Cat row1,row2,row3

表示 Cat 在行 row1、row2、row3 都出現過，這樣通過查找單詞所對應的行的 rowid 就可以迅速找到文句記錄。

索引建好後，我們可以在該使用者下查到 Oracle 自動產生了以下幾個表（假設索引名為 myindex）：DR$myindex$I、DR$myindex$K、DR$myindex$R、DR$myindex$N 其中以 I 表最重要，可以查詢一下該表，看看有什麼內容：

```
SQL> CREATE TABLE mytext (text VARCHAR2(100));
SQL> INSERT INTO mytext
    VALUES ('I Love www.itpub.net');
SQL> COMMIT;
SQL> CREATE INDEX mytext_idx ON mytext(text) INDEXTYPE
IS ctxsys.CONTEXT;
SQL> SELECT token_text, token_count FROM dr$mytext_idx$i;

TOKEN_TEXT                TOKEN_COUNT
I                                1
ITPUB                            1
LOVE                             1
NET                              1
WWW                              1
```

注意 TOKEN_TEXT 裡面字元全部是大寫的，預設情況下全文索引是不區分大小寫的.

	SQL> DESC dr$mytext_idx$i; 名稱 是否為空 類型 TOKEN_TEXT NOT NULL VARCHAR2(64) TOKEN_TYPE NOT NULL NUMBER(3) TOKEN_FIRST NOT NULL NUMBER(10) TOKEN_LAST NOT NULL NUMBER(10) TOKEN_COUNT NOT NULL NUMBER(10) TOKEN_INFO BLOB 可以看到，該表中保存的其實就是 Oracle 分析你的文檔後，生成的 token 記錄在這裡，包括 token 出現的位置、次數、hash 值等。
建立索引	語法如下，索引建立好後可以用 ConTains 進行查詢。 CREATE INDEX [*schema*.]*index* on [*schema*.]*table*(*column*) INDEXTYPE IS ctxsys.context [ONLINE] LOCAL [(PARTITION [*partition*] [PARAMETERS('*paramstring*')] [, PARTITION [*partition*] [PARAMETERS('*paramstring*')]])] [PARAMETERS(*paramstring*)] [PARALLEL n] [UNUSABLE];
全文檢索	1. 設置詞法分析器(lexer) Oracle 實現全文檢索，其機制其實很簡單。即通過 Oracle 專利的詞法分析器(lexer)，將文章中所有的表意單元（Oracle 稱為 term）找出來，記錄在一組 以 dr$開頭的表中，同時記下該 term 出現的位置、次數、hash 值等資訊。檢索時，Oracle 從這組表中查找相應的 term，並計算其出現頻率，根據某個演算法來計算每個文檔

	的得分（score）,即所謂的‘匹配率’。而 lexer 則是該機制的核心，它決定了全文檢索的效率。 2. Oracle 針對不同的語言提供了不同的 lexer, 而我們通常能用到其中的三個： a. basic_lexer： 針對英語。它能根據空格和標點來將英語單詞從句子中分離，還能自動將一些出現頻率過高已經失去檢索意義的單詞作為‘垃圾’處理，如 if , is 等，具有較高的處理效率。但該 lexer 應用於漢語則有很多問題，由於它只認空格和標點，而漢語的一句話中通常不會有空格，因此，它會把整句話作為一個 term,事實上失去檢索能力。 b. chinese_vgram_lexer： 專門的漢語分析器，支持所有中文字元集（ZHS16CGB231280 ZHS16GBK ZHT32EUC ZHT16BIG5 ZHT32TRIS ZHT16MSWIN950 ZHT16HKSCS UTF8 ）。該分析器按字為單元來分析漢語句子。 c. chinese_lexer: 這是一個新的漢語分析器，只支持 utf8 字元集。上面已經看到，chinese vgram lexer 這個分析器由於不認識常用的漢語詞彙，因此分析的單元非常機械，因此常會對分析 term 形成沒有意義的，反而影響效率。chinese_lexer 的最大改進就是該分析器 能認識大部分常用漢語詞彙，因此能更有效率地分析句子。但是它只支援 utf8, 如果你的資料庫是 zhs16gbk 字元集，則只能

	使用 Chinese vgram lexer.
	d. 如果不做任何設置，Oracle 可以使用 basic_lexer 這個分析器，要指定使用哪一個 lexer 如下操作：
	在 ctxsys 用戶下建立一個 preference:
	BEGIN
	ctx_ddl.create_preference ('my_lexer', 'chinese_vgram_lexer');
	END;
	
	在建立 intermedia 索引時，指明所用的 lexer：
	CREATE INDEX myindex ON mytable(mycolumn) indextype is ctxsys.context
	parameters('lexer my_lexer');
	
	這樣建立的全文檢索索引，就會使用 chinese_vgram_lexer 作為分析器。 相應的，索引中文就比索引英文佔用的表空間多了許多。Oracle Text 為了性能不得不犧牲了空間。

檢索 範例一	檢索操作 CONTAINS ： 1. SELECT 語句中，可以在 WHERE 指定 CONTAINS 操作指令。還可以指定返回記錄的得分（SCORE） SELECT score (1) title FROM news WHERE contains (text, 'Oracle', 1) > 0; 2. 得分 SCORE 是指查詢結果的貼切程度。得分越高表示查詢

	資訊滿意度越高，可以根據 SCORE 進行排序。 SELECT score (1), title FROM news WHERE contains (text, 'Oracle', 1) > 0 ORDER BY score (1) DESC; SELECT score (1), title, issue_date FROM news WHERE contains (text, 'Oracle', 1) > 0 AND issue_date >= ('01-OCT-97') ORDER BY score (1) DESC; 3. 根據 contains 中的不同數位標示各個 contains 返回的分數 Score SELECT id,score(1),score (2),score (1)+ score (2) total, text FROM mytable WHERE contains (text, 'biti', 1) > 0 OR contains (text, 'hello', 2) > 0 ORDER BY total DESC;
檢索 範例二	使用的帳號為 db_test 使用系統管理者登入，並且授於 db_test 相關的權限 GRANT "CTXAPP" TO "db_test"; ALTER USER "db_test" DEFAULT ROLE ALL;

GRANT EXECUTE ON "CTXSYS"."CTX_ADM" TO "db_test";

GRANT EXECUTE ON "CTXSYS"."CTX_CATSEARCH" TO "db_test";

GRANT EXECUTE ON "CTXSYS"."CTX_CONTAINS" TO "db_test";

GRANT EXECUTE ON "CTXSYS"."CTX_DDL" TO "db_test";

GRANT EXECUTE ON "CTXSYS"."CTX_DOC" TO "db_test";

GRANT EXECUTE ON "CTXSYS"."CTX_QUERY" TO "db_test";

GRANT EXECUTE ON "CTXSYS"."CTX_ULEXER" TO "db_test";

GRANT EXECUTE ON "CTXSYS"."CTX_XPCONTAINS" TO "db_test";

採用 chinese_lexer

SQL> begin ctx_ddl.create_preference('new_lexer','chinese_lexer');
end;

/*

PL/SQL procedure successfully completed.

這樣子就建立了一個 object 了

再來建立 index

SQL> create index km_t_data_user_id_tqindex<index 名字> on t_data<那個 table>(user_id<欄位名字>) indextype is ctxsys.context parameters('LEXER db_test.new_lexer');<使用者名字.建立的 object>

	Index created. 刪除 index drop index name_index; 索引同步 exec ctx_ddl.sync_index('<index 名字>'); 查尋用法 SQL> select subject from t_data where contains(subject,'oracle',1)>0;
官方網站	http://www.oracle.com/global/tw/index.html

4-2-2 MySQL

簡介	1. MySQL 從 3.23.23 版開始支援全文索引和全文檢索。 2. 在 MySQL 中，全文索引的索引類型為 FULLTEXT。 3. 全文索引可以在 VARCHAR 或者 TEXT 類型的列上建立。它可以通過 CREATE TABLE 命令建立，也可以通過 ALTER TABLE 或 CREATE INDEX 命令建立。 4. 對於大規模的資料集，通過 ALTER TABLE（或者 CREATE INDEX）命令建立全文索引要比把記錄插入帶有全文索引的空表更快。 5. MySQL 索引是以 B-Tree 方式儲存，index 可以是一個欄位或是由多個欄位複合而成。
全文檢索	建立可以全文檢索的欄位資料表 CREATE TABLE testft (testint int, testvc varchar(50), testtxt text, FULLTEXT

	(testvc, testtxt)); 語法： SELECT [columns] FROM [table] WHERE MATCH ([indexed_columns]) AGAINST (‘keyword’) 範例： SELECT vehicle_id, description FROM New_Vehicles WHERE MATCH (description) AGAINST (‘options’); 預設全文檢索關鍵字長度為 4 word。 可以修改參數來調整 my.cnf : [mysqld] ft_min_word_len=3

範例	<?php
	/**
	* DO NOT CHANGE
	*/
	if (empty($lang) \|\| !is_array($lang))
	{
	$lang = array();
	}
	
	// DEVELOPERS PLEASE NOTE
	//
	// All language files should use UTF-8 as their encoding and the files

	must not contain a BOM.
	//
	// Placeholders can now contain order information, e.g. instead of
	// 'Page %s of %s' you can (and should) write 'Page %1$s of %2$s', this allows
	// translators to re-order the output of data while ensuring it remains correct
	//
	// You do not need this where single placeholders are used, e.g. 'Message %d' is fine
	// equally where a string contains only two placeholders which are used to wrap text
	// in a url you again do not need to specify an order e.g., 'Click %sHERE%s' is fine
	
	=> '這裡可以管理內容索引. 因為一般只使用一個後台, 您可以刪除所有不用的索引. 在改變搜尋設定(例如最小/最大字元串長度)後, 建議重新產生索引以體現修改.',
	$lang = array_merge($lang, array(
	'ACP_SEARCH_INDEX_EXPLAIN'
	
	=> '這裡您可以設定如何使用搜尋和檢索文章. 您可以設定選項限制搜尋動作的處理器負載. 一部分設定和和搜尋引擎的設定是一樣的.'
	'ACP_SEARCH_SETTINGS_EXPLAIN',

	=> '詞頻閾值', 'COMMON_WORD_THRESHOLD' => '在所有的文章中都頻繁出現的單詞將被識別為高頻詞. 高頻詞在搜尋中將被忽略. 設定 0 則取消這項功能. 只有當文章數大於 100 時這個設定才能生效.', 'COMMON_WORD_THRESHOLD_EXPLAIN' => '您確認更換搜尋後端嗎? 更換後您需要重新建立索引. 如果您不打算再切換回舊的後端, 您可以刪除原先的索引以釋放空間.', 'CONFIRM_SEARCH_BACKEND' => '繼續前次的索引刪除', 'CONTINUE_DELETING_INDEX' => '存在一個已經開始的索引刪除. 要瀏覽索引頁面您必須先完成它.', 'CONTINUE_DELETING_INDEX_EXPLAIN' => '繼續前次的索引產生', 'CONTINUE_INDEXING' => '存在一個已經開始的索引產生. 要瀏覽索引頁面您必須先完成它.',

	'CONTINUE_INDEXING_EXPLAIN' 'CREATE_INDEX' => '建立索引', 'DELETE_INDEX' => '刪除索引', => '刪除索引進行中', 'DELETING_INDEX_IN_PROGRESS' => '搜尋後端正在清除索引，這需要幾分鐘的時間.', 'DELETING_INDEX_IN_PROGRESS_EXPLAIN' => 'MySQL 全文檢索後端只能在 MySQL4 或更高的版本中使用.', 'FULLTEXT_MYSQL_INCOMPATIBLE_VERSION' => 'MySQL 全文檢索只能在 MyISAM 格式資料表中使用.', 'FULLTEXT_MYSQL_NOT_MYISAM' => '索引文章總數', 'FULLTEXT_MYSQL_TOTAL_POSTS' => '支援非拉丁 UTF-8 字元使用 mbstring:', 'FULLTEXT_MYSQL_MBSTRING' => '支援非拉丁 UTF-8 字元使用 PCRE:', 'FULLTEXT_MYSQL_PCRE'

	=> '如果 PCRE 沒有 unicode 字元屬性, 搜尋後端會嘗試使用 mbstring 的正則表達式機制.', 'FULLTEXT_MYSQL_MBSTRING_EXPLAIN' => '這個搜尋後端需要使用 PCRE unicode 字元屬性, 這只在 PHP 版本 4.4, 5.1 或更高中可用, 如果您需要搜尋非拉丁字元.', 'FULLTEXT_MYSQL_PCRE_EXPLAIN' => '綜合搜尋設定', 'GENERAL_SEARCH_SETTINGS' => '前往索引頁面', 'GO_TO_SEARCH_INDEX' => '索引統計', 'INDEX_STATS' => '索引進行中', 'INDEXING_IN_PROGRESS' => '搜尋後端正在檢索討論區的所有文章. 取決於討論區資料量的大小, 這可能需要幾分鐘的時間.', 'INDEXING_IN_PROGRESS_EXPLAIN'

	=> '搜尋頁面系統負載限制', 'LIMIT_SEARCH_LOAD' => '如果一分鐘內系統負載超過這個值，搜尋頁面將無法使用，1.0 約等於 100% 的處理器負載. 這個功能只在基於 UNIX/Linux 系統的伺服器上有效.', 'LIMIT_SEARCH_LOAD_EXPLAIN' => '索引的最大字元長度', 'MAX_SEARCH_CHARS' => '大於這個長度的短語將不會被檢索.', 'MAX_SEARCH_CHARS_EXPLAIN' => '索引的最小字元長度', 'MIN_SEARCH_CHARS' => '小於這個長度的短語將不會被檢索.', 'MIN_SEARCH_CHARS_EXPLAIN' => '最小帳號長度', 'MIN_SEARCH_AUTHOR_CHARS' => '會員在搜尋作者姓名時必須輸入的字元串長度. 如果作者帳號

	長度小於這數量字, 您一人可以輸入完整帳號搜尋會員的文章.', 'MIN_SEARCH_AUTHOR_CHARS_EXPLAIN' => '進度條', 'PROGRESS_BAR' => '訪客搜尋間隔', 'SEARCH_GUEST_INTERVAL' => '在多次搜尋中訪客必須等待的間隔時間(秒).', 'SEARCH_GUEST_INTERVAL_EXPLAIN' => '所有 id 不高於 %1$d 的文章都已經建立索引, 這一操作中處理了 %2$d 個文章. 速度接近 %3$.1f 文章每秒. 索引正在進行中…', 'SEARCH_INDEX_CREATE_REDIRECT' => '所有 id 不高於 %1$d 的文章都已經從搜尋索引中刪除. 刪除正在進行中…', 'SEARCH_INDEX_DELETE_REDIRECT' => '已經對資料庫中的所有文章建立了索引.', 'SEARCH_INDEX_CREATED' => '成功清除了這個後端的索引資料.',

	'SEARCH_INDEX_REMOVED' => '會員搜尋間隔', 'SEARCH_INTERVAL' => '會員搜尋的最短間隔時間(秒).', 'SEARCH_INTERVAL_EXPLAIN' => '搜尋結果快取時間', 'SEARCH_STORE_RESULTS' => '快取的結果在這個時間(秒)後將失效. 設定為 0 則停用快取.', 'SEARCH_STORE_RESULTS_EXPLAIN' => '搜尋後端', 'SEARCH_TYPE' => 'phpBB 允許您選擇用於全文檢索的後端類型. 預設後端是 phpBB 自帶的全文檢索.', 'SEARCH_TYPE_EXPLAIN' => '您已經更換了檢索後端. 要使用新的後端, 您必須確認新的後端包含建好的索引.', 'SWITCHED_SEARCH_BACKEND' => '已檢索關鍵詞總數',

	'TOTAL_WORDS' => '與文章索引關聯的關鍵詞總數', 'TOTAL_MATCHES' => '啟用檢索功能', 'YES_SEARCH' => '允許會員使用搜尋功能，包括會員搜尋.', 'YES_SEARCH_EXPLAIN' => '啟用全文更新', 'YES_SEARCH_UPDATE' => '當文章發表時更新全文索引，如果停用檢索這個功能將停用.', 'YES_SEARCH_UPDATE_EXPLAIN')); ?>
官方網站	http://www.mysql.com/

5. 本專案使用案例 - SQL Server 2008 全文檢索搜尋

5-1 SQL Server2008 全文檢索簡介

SQL Server 2008 提供了一項功能，可讓應用程式和使用者針對 SQL Server 資料表中以字元為基礎的資料發出全文檢索查詢。不過資料庫管理員（Database

Administrator）必須先在特定的資料表上建立全文檢索索引，然後才能在此資料表上執行全文檢索查詢。全文檢索索引包括資料表中一或多個以字元為基礎的資料行，這些資料行可以具有下列任何資料類型：char、varchar、nchar、nvarchar、text、ntext、image、xml、varbinary 或 varbinary(max)。每個全文檢索索引都會為基底資料表中的一或多個資料行建立索引，而且每個資料行都可以具有特定的語言。以 SQL Server 2008 為例，全文檢索搜尋以可支援 50 種以上不同的語言，例如：英文、西班牙文、中文、日文、阿拉伯文、孟加拉文和印度文等等[1]。

SQL Server 2008 會針對每種支援的語言提供語言特有的語言元件，包括斷詞工具、字幹分析器和空的同義字檔案。SQL Server 2008 也會針對每種全文檢索語言提供一個檔案，讓使用者可以在其中選擇性地定義語言特有的同義字，以便擴充搜尋查詢的範圍，該檔案稱為「同義字檔案」（Thesaurus File）；此外，也提供了「系統停用字詞表」來支援特定語言或商務案例。也就是說，可以透過加入和移除停用字詞 (也稱為非搜尋字) 來適應不同的電子商務全文檢索狀況。

此外，為了撰寫全文檢索查詢，SQL Server 2008 提供了一組全文檢索述詞（CONTAINS 和 FREETEXT）與資料列集值函數（CONTAINSTABLE 和 FREETEXTTABLE）。應用程式和使用者可以使用這些項目來執行各種全文檢索搜尋類型，例如：搜尋單一字詞或片語（並選擇性地排序結果集等級）、搜尋接近其他字詞或片語的字詞或片語，或是搜尋特定字詞的同義字變化。

5-2 SQL Server2008 全文檢索搜尋運作類型

全文檢索查詢會根據特定語言的規則 (例如英文或日文) 在單字與片語上運作，藉以針對全文檢索索引中的文字資料執行語言搜尋。全文檢索查詢可以包含簡單的單字和片語，或者單字或片語的多種形式，因此全文檢索搜尋技術特別

[1] 請參閱＜http://msdn.microsoft.com/zh-tw/library/ms176076.aspx＞

適用於各種商務案例，例如：搜尋電子商務網站上的商品或文件、搜尋律師事務所法律資料儲存檔案中的個案記錄)，或搜尋人力資源部門中比對工作描述與預存的履歷表等等。不論商務案例為何，全文檢索搜尋的基本管理和開發工作都是相同的。只不過在不同的商務案例上，可能會進一步調整全文檢索索引和查詢來符合商務目標。例如，對於電子商務而言，發揮最佳效能可能會比排序結果等級、重新叫用（Recall）精確度 (全文檢索查詢實際傳回的現有相符項目數) 或支援多國語言更重要；對於律師事務所而言，傳回每個可能的資訊可能是最重要的考量。

5-3 SQL Server2008 全文檢索搜尋在資料庫的設定

在任何的商務案例上對於全文檢索搜尋的設定，資料庫管理員都會執行以下資料庫中的資料表資料行的設定。基本步驟如下：

1. 建立全文檢索目錄。
2. 在計畫搜尋的每個資料表上，建立全文檢索索引：
    a. 識別想要包含在全文檢索索引中的每個文字資料行。
    b. 如果給定的資料行包含儲存成二進位資料 (varbinary、varbinary(max) 或 image 資料) 的文件，就必須指定資料表資料行 (「類型資料行」(Type Column))，以便識別要進行索引之資料行中每份文件的類型。
    c. 指定想讓全文檢索搜尋用於資料行中文件的語言。
    d. 選擇想要針對全文檢索索引使用的變更追蹤機制，以便追蹤基底資料表及其資料行中的變更。

設定完成後全文檢索搜尋會透過使用「語言元件」(Linguistic Component)，包括：斷詞工具和字幹分析器、停用字詞表 (也稱為非搜尋字詞表) ，以及同義字檔案

等支援多國語言。其中同義字檔案會支援所有使用對應語言的全文檢索索引，而停用字詞表也可以與任意數目的全文檢索索引相關聯。

5-4 SQL Server2008 全文檢索的查詢

在資料行已經加入至全文檢索索引之後，應用程式和使用者就可以針對資料行中的文字執行全文檢索查詢。這些查詢可以搜尋下列的任何項目：

- 一或多個特定的單字或片語 (不可分割的詞彙（Simple Term）)
- 以指定之文字開頭的單字或片語 (前置詞彙（Prefix Term）)
- 特定單字的字形變化 (衍生詞彙（Generation Term）)
- 靠近另一個單字或片語的單字或片語 (相近詞彙（Proximity Term）)
- 特定單字的同義字變化 (同義字（Thesaurus）)
- 使用加權值的單字或片語 (加權詞彙（Weighted Term）)

此外，在全文檢索查詢會使用一小組 Transact-SQL 述詞 (CONTAINS 和 FREETEXT) 與函數 (CONTAINSTABLE 和 FREETEXTTABLE)。例如：

- 電子商務 - 搜尋網站上的產品：

  SELECT product_id FROM products WHERE
  CONTAINS(product_description, ”Snap Happy 100EZ” OR
  FORMSOF(THESAURUS,’Snap Happy’) OR ‘100EZ’) AND
  product_cost<200 …

- 人員招募案例 - 搜尋具有 SQL Server 使用經驗的工作應徵者：

SELECT candidate_name, SSN FROM candidates WHERE CONTAINS(candidate_resume,”SQL Server”) AND candidate_division =DBA

相關敘述詞與函數會在後面小節進一步介紹。

5-5 SQL Server2008 全文檢索搜尋架構

全文檢索搜尋是由全文檢索引擎所提供，此全文檢索引擎扮演兩個角色，分別是索引支援和查詢支援。以 SQL Server 2008 為例，全文檢索搜尋架構是由兩個處理程式所組成：

- SQL Server 處理序 (sqlservr.exe)
- 篩選背景程式主機處理序 (fdhost.exe)

伺服器執行個體會針對所有多執行緒篩選使用多執行緒處理序，而針對所有單一執行緒篩選使用單一執行緒處理序。此外，FDHOST 啟動器服務 (MSSQLFD Launcher) 會建立 fdhost.exe 處理序，而且這些處理序會在 FDHOST 啟動器服務帳戶的安全性認證底下執行。透過執行 FDHOST 這個服務，才能讓全文檢索索引和全文檢索查詢運作。整個處理序所包含全文檢索搜尋架構的元件與其關聯性如下圖所示，並說明如後。

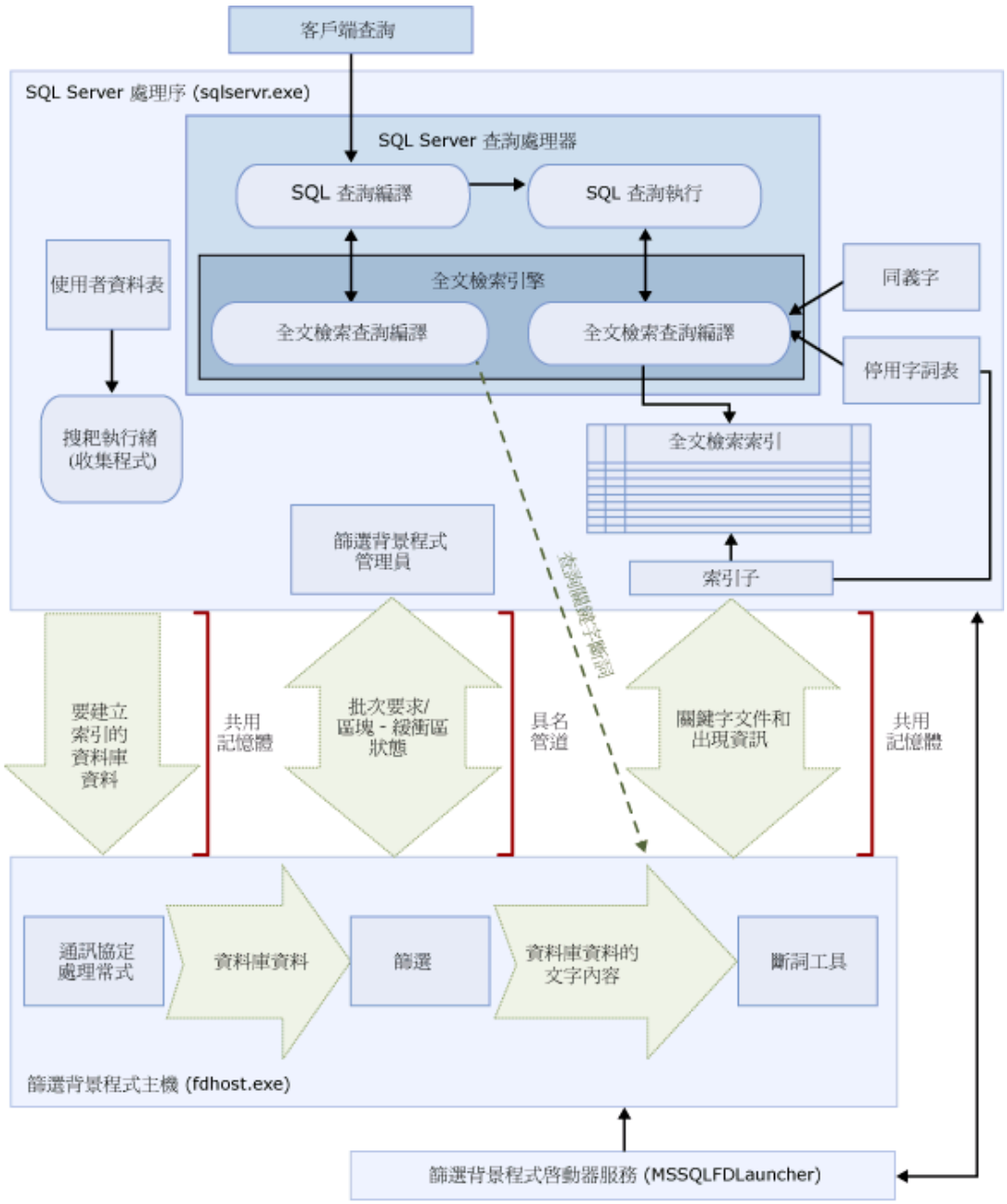


圖 2　全文檢索搜尋架構

a.　SQL Server 處理序 (sqlservr.exe)

全文檢索搜尋會使用 SQL Server 處理序的下列元件：

- 使用者資料表：

  這些資料表包含要進行全文檢索索引的資料。

- 全文檢索收集程式：

  全文檢索收集程式會使用全文檢索搜耙（也就是全文檢索擴展）執行緒。此元件負責排程和驅動全文檢索索引母體擴展，以及監視全文檢索目錄[2]。

- 同義字檔案：

  這些檔案包含搜尋詞彙的同義字。

- 停用字詞表物件：

  停用字詞表物件包含對搜尋沒有任何幫助之常見單字的清單。

- SQL Server 查詢處理器：

  查詢處理器會編譯並執行 SQL 查詢。如果某個 SQL 查詢包含全文檢索搜尋查詢，該查詢就會在編譯和執行期間傳送至全文檢索引擎。系統會針對全文檢索索引比對查詢結果。

- 全文檢索引擎：

  SQL Server 中的全文檢索引擎與查詢處理器完全整合，全文檢索引擎會編譯並執行全文檢索查詢。在查詢執行期間，全文檢索引擎可能會收到來自同義字和停用字詞表的輸入。在 SQL Server 2008 和更新版本中，Full-Text Engine for SQL Server 會在 SQL Server 查詢處理器內部執行。

- 索引寫入器 (索引子)：

  索引寫入器會建立用來儲存索引 Token 的結構。

- 篩選背景程式管理員：

  篩選背景程式管理員會負責監視全文檢索引擎篩選背景程式主機的狀態。

b. 篩選背景程式主機處理序 (fdhost.exe)

[2] 從 SQL Server 2008 開始，全文檢索目錄是虛擬物件，而且不屬於任何檔案群組，且是參考一組全文檢索索引的邏輯概念。

篩選背景程式主機是全文檢索引擎所啟動的處理序。它會執行下列的全文檢索搜尋元件，而這些元件會負責存取、篩選和斷詞處理資料表的資料，以及斷詞處理和詞幹分析查詢輸入。篩選背景程式主機的元件功能說明如下：

- 通訊協定處理常式：

  這個元件會從記憶體中提取資料以便進一步處理，而且會從指定之資料庫中的使用者資料表中存取資料。其中一項責任就是從建立全文檢索索引的資料行中蒐集資料，並將資料傳遞給篩選背景程式主機，然後此處理序將會視需要套用篩選和斷詞工具。

- 篩選工具：

  某些資料類型需要先篩選，然後才能針對文件中的資料建立全文檢索索引，包括 varbinary、varbinary(max)、image 或 xml 資料行中的資料。用於給定文件的篩選會因其文件類型而不同。例如，Microsoft Word (.doc) 文件、Microsoft Excel (.xls) 文件和 XML (.xml) 文件會使用不同的篩選。然後，篩選會從文件中擷取文字區塊，並且移除內嵌的格式，並保留文字和文字位置的相關資訊，其結果就是文字資訊的資料流。

- 斷詞工具和字幹分析器：

  斷詞工具是一項語言特有的元件，它會根據給定語言的語彙規則來尋找文字分界 —「斷詞」(Word Breaking))。每個斷詞工具都與語言特有的字幹分析器元件相關聯，而且此元件會進行動詞變化和執行字形擴展。建立索引時，篩選背景程式主機會使用斷詞工具和字幹分析器，針對來自給定資料表資料行的文字資料執行語言分析。與全文檢索索引中資料表資料行相關聯的語言會決定哪些斷詞工具和字幹分析器要用於建立該資料行的索引。

5-6 SQL Server2008 全文檢索索引和查詢處理程序

全文檢索搜尋的索引元件會負責全文檢索索引的初始擴展，以及修改過全文檢索索引資料表中的資料後續更新。因此包括「全文檢索索引處理程序」，以及「全文檢索查詢處理程序」兩個動作。

a. 全文檢索索引處理程序

全文檢索擴展 (也就是搜耙) 起始時，全文檢索引擎會將大批的資料發送至記憶體中，並通知篩選背景程式主機。此主機會針對資料進行篩選並斷詞，並且將轉換的資料轉換成反向字詞清單。然後，全文檢索搜尋會從這些字詞清單中提取轉換的資料、處理資料以便移除停用字詞，並且將批次的字詞清單保存在一或多個反向索引中。對 varbinary(max)或 image 資料行中儲存的資料編製索引時，執行 IFilter 介面的篩選會依據為該資料所指定的檔案格式(例如 Microsoft Word 格式) 擷取文字。收集的文字資料在經由文字分隔的處理之後，會分隔成 Token 或關鍵字。用於 Token 化的語言是在資料行層級指定，也可由篩選元件在 varbinary(max)、image 或 xml 資料中識別。完成全文檢索擴展後會觸發最後的合併程序，將索引片段合併成一個主要的全文檢索索引。如此可提升查詢的效能，因為只需要查詢一個主索引而不需查詢數個索引片段，而且可使用較佳的計分系統來排定索引關聯順序。

b. 全文檢索查詢處理程序

查詢處理器會將查詢的全文檢索部分傳遞至全文檢索引擎，以便進行處理。全文檢索引擎會執行斷詞並選擇性地執行同義字展開、詞幹分析和停用字詞(非搜尋字) 處理。然後，查詢的全文檢索部分會以 SQL 運算子的形式表示，主要表示成資料流資料表值函數 (STVF)。在查詢執行期間，這些 STVF 會存取反向索引來擷取正確的結果。接著，這些結果會在此時傳回用戶端，或在傳回用戶端之前進一步處理。

5-7 SQL Server2008 全文檢索引擎

SQL Server 全文檢索引擎是一個全文檢索索引與搜尋引擎。在 SQL Server 2008 中，全文檢索引擎已經完全整合至 Database Engine 中。全文檢索引擎在位於 SQL Server 處理序 (Sqlservr.exe) 中，而非位於個別的處理序 (Msftesql.exe) 中。將全文檢索引擎整合至 Database Engine 中可以改善全文檢索管理能力、混合式查詢的最佳化，以及整體效能。全文檢索引擎負責對全文檢索索引的讀取和寫入，該索引現在是儲存在 SQL Server 中。全文檢索引擎支援索引與查詢兩項作業，說明如下：

- 索引：

  當資料經過篩選而且文字經過斷詞處理之後，SQL Server 處理序就會接收結果並建立它們的索引。

- 查詢：

  全文檢索引擎會處理全文檢索搜尋查詢，並且判斷基底資料表中的哪些項目 (資料列或文件) 符合全文檢索選取準則。

a. 設定全文檢索語言元件

以 SQL Server 2008 為例，全文檢索搜尋支援將近 50 種不同的語言，例如：英文、西班牙文、中文、日文、阿拉伯文、孟加拉文和印度文。全文檢索索引所包含的每個資料行都與 Microsoft Windows 地區設定識別碼 (LCID) 相關聯，而這個識別碼就等於全文檢索搜尋所支援的語言。例如，LCID 1033 等於美式英文，而 LCID 2057 等於英式英文。SQL Server 針對每個支援的全文檢索語言提供了一些語言特有元件來支援索引和查詢使用該語言所儲存的全文檢索資料。

語言特有元件包括「斷詞工具」 (Word Breaking)和「字幹分析器」(Stemmer)。斷詞工具會根據給定語言的語彙規則來尋找文字分界。每個斷詞工具都與針對相同語言進行動詞變化的字幹分析器相關聯。斷詞工具 (和字幹分析器) 與篩選會在篩選背景程式主機處理序 (fdhost.exe) 中執行。此外，從 SQL Server 2008 也提供了包含基本「停用字詞」（Stopword，也稱為非搜尋字) 集合的系統停用字詞表來忽略無助於全文檢索查詢搜尋的單字。以英文地區（LCID 1033 或 LCID 2057）設定為例，"a"、"and"、"is" 和 "the" 都會被視為停用字詞。一般而言，會根據查詢需求設定一或多個同義字檔案和停用字詞表。

SQL Server 也會針對每個全文檢索語言安裝「同義字」 (Thesaurus)檔案[3]，以及全域同義字檔案。剛開始安裝的同義字檔案基本上是空白的，但是可以透過編輯來定義特定語言或商務狀況的同義字 (Synonym)。透過持續維護符合全文檢索資料的同義字檔案可以有效地擴大進行全文檢索查詢的範圍。不過在 varbinary、varbinary(max)、image 或 xml 資料類型資料行中索引文件需要執行額外處理的篩選，以 SQL Server 為例，這些文件類型可能是.doc、.pdf、.xls 和 .xml 等等。

b. 斷詞工具(Word Breaking)**與字幹分析器**(Stemmer)

斷詞工具(Word Breaker)及字幹分析器(Stemmer)，會在所有全文檢索索引資料上執行語文分析。語文分析包括找出文字分界 (斷詞) 以及動詞變化 (字根處理)。斷詞工具與字幹分析器是語言特有的工具，而且語言分析的規則會因不同的語言而有所差異。對於給定的語言而言，斷詞工具會根據語言的語彙規則，判斷文字分界存在的位置，藉以識別個別單字。每個"單字" (也稱為 Token) 都會使用壓縮表示來插入全文檢索索引中，以便減少其大小。「字幹分析器」(Stemmer)

[3] 指的是個別語言的同義字檔案

會根據該語言的規則來產生特定單字的字形變化 (例如，"running"、"ran" 和 "runner" 是 "run" 單字的不同形態)。

使用語言特有的斷詞工具會使得針對該語言產生的詞彙更正確。如果有語系的斷詞工具，但沒有特定次語言的斷詞工具，則會使用主要語言。例如，處理加拿大法文時會使用法文文字分隔。如果某種特定語文沒有文字分隔，則會使用中性文字分隔。使用中性文字分隔時，會以中性字元來中斷文字，例如，空白與標點符號。

5-8 SQL Server2008 建立與維護全文檢索索引 - 母體擴展（Population）

全文檢索引擎會使用全文檢索索引中的資訊來編譯全文檢索查詢，以便快速地在資料表中搜尋特定單字或單字組合。全文檢索索引會儲存重要單字及這些單字在資料庫 資料表之一或多個資料行內位置的相關資訊。全文檢索索引是一種特殊類型的 Token 式功能索引，由 Full-Text Engine for SQL Server 所建立與維護。建立全文檢索索引的程序與建立其他索引類型的程序大不相同。全文檢索引擎會根據個別 Token 從索引中的文字建立反向、堆疊以及壓縮的索引結構，而不是根據特定資料列中所儲存的值來建構 B 型樹狀結構。在 SQL Server 2008 中，全文檢索索引的大小只受限於執行 SQL Server 執行個體之電腦的可用記憶體資源。

在每個資料表只允許有一個全文檢索索引。若要對資料表建立全文檢索索引，該資料表必須有單一的非 Null 唯一資料行。您可以針對 char、varchar、nchar、nvarchar、text、ntext、image、xml、varbinary 和 varbinary(max) 類型的資料行建立全文檢索索引，並且建立全文檢索搜尋的索引。針對 image、varbinary 或 varbinary(max) 建立全文檢索索引會要求您指定類型資料行。類型資料行是

一個資料表資料行，可以在每個資料列中儲存文件的副檔名 (.doc、.pdf 和 .xls 等等)。

建立與維護全文檢索索引的程序稱為「母體擴展」(Population) (也稱為「搜耙」(Crawl))。全文檢索索引母體擴展有三種類型：完整母體擴展、以變更追蹤為基礎的母體擴展，以及以時間戳記為基礎的累加母體擴展。建立、更改與卸除全文索引的語法如下：

若要建立全文檢索索引

- CREATE FULLTEXT INDEX (Transact-SQL)

若要更改全文檢索索引

- ALTER FULLTEXT INDEX (Transact-SQL)

若要卸除全文檢索索引

- DROP FULLTEXT INDEX (Transact-SQL)

5-9 使用全文檢索搜尋查詢 SQL Server

SQL Server 會提供一套全文檢索述詞 (CONTAINS 和 FREETEXT) 與資料列集值函數 (CONTAINSTABLE 和 FREETEXTTABLE)來撰寫全文檢索查詢。全文檢索查詢會使用全文檢索述詞 (CONTAINS 和 FREETEXT) 與函數 (CONTAINSTABLE 和 FREETEXTTABLE)。這些述詞支援豐富的 Transact-SQL 語法，而這種語法支援各種形式的查詢詞彙，因此若要撰寫全文檢索查詢，必須了解使用這些述詞與函數的時機和方式。以下列出一些述詞與函數，並且討論 CONTAINS 述詞與 CONTAINSTABLE 函數之間的共同點與不同點，並且描述如何使用它們執行不同的搜尋類型。(以下會說明如何微調和最佳化全文檢索查

詢。)

5-10 全文檢索述詞(CONTAINS 和 FREETEXT)

CONTAINS 和 FREETEXT 都是在 SELECT 陳述式的 WHERE 或 HAVING 子句中指定的。它們可與任何其他 Transact-SQL 述詞結合，例如 LIKE 和 BETWEEN。CONTAINS 和 FREETEXT 述詞會傳回 TRUE 或 FALSE 值，並用來指定選取準則以便判斷給定的資料列是否符合全文檢索查詢。符合的資料列就會傳回結果集中。使用 CONTAINS 或 FREETEXT 時，可以指定要搜尋資料表中的單一資料行、資料行清單或所有資料行。此外也可以針對斷詞和詞幹分析、同義字查閱以及非搜尋字移除，指定給定全文檢索查詢將使用之資源的語言。

CONTAINS 和 FREETEXT 適用於不同種類的比對，如下所示：

- 使用 CONTAINS (或 CONTAINSTABLE) 以進行下列各種比對：單字和片語的精確或模糊 (較不精確) 比對、單字彼此在一定距離之間的接近度，或加權相符。使用 CONTAINS 時，至少必須指定一個要搜尋之文字的搜尋條件，以及判斷是否相符的條件。同時也可以在搜尋條件之間使用邏輯作業（AND、OR、AND NOT）。
- FREETEXT (或 FREETEXTTABLE) 是用來比對指定之單字、片語或句子「Freetext 字串」的意義，但不比對確切的用字。如果在指定之資料行的全文檢索索引中找到任何詞彙或任何形式的詞彙，就會產生相符項目。

5-11 範例 1 - 搭配 <simple_term> 使用 CONTAINS

下列範例會尋找所有價格是 $80.99，且含有 "Mountain" 這個單字的產品。

```
USE AdventureWorks;
GO
```

```
SELECT Name, ListPrice
FROM Production.Product
WHERE ListPrice = 80.99；simple_term
    AND CONTAINS(Name, 'Mountain');
GO
```

5-12 範例 2 -使用 FREETEXT 搜尋含有指定字元值的單字

下列範例會搜尋包含 vital、safety 和 components 相關單字的所有文件。

```
USE AdventureWorks;
GO
SELECT Title
FROM Production.Document
WHERE FREETEXT (Document, 'vital safety components' );
GO
```

5-13 全文檢索函數 (CONTAINSTABLE 和 FREETEXTTABLE)

CONTAINSTABLE 和 FREETEXTTABLE 函數的參考方式就如同 SELECT 陳述式之 FROM 子句中的一般資料表名稱。它們會傳回符合全文檢索查詢之零、一或多個資料列的資料表。傳回的資料表僅包含基底資料表的資料列，而這些資料列符合函數之全文檢索搜尋條件中指定的選取準則。

使用其中一個函數的查詢會針對每個資料列傳回一個相關次序值 (RANK) 和全文檢索索引鍵 (KEY)。說明如下：

- KEY 資料行：

  KEY 資料行會傳回所傳回之資料列的唯一值；KEY 資料也可用來指定選取準則。

- RANK 資料行：

  RANK 資料行會傳回每個資料列的「等級值」(Rank Value)，表示資料列與選取準則的符合程度。資料列中文字或文件的等級值越高，該資料列與給定全文檢索查詢的關聯性就越大。值得注意的是，不同的資料列可能會以完全相同的方式排序等級。另外可以透過指定選擇性 top_n_by_rank 參數，來限制要傳回的相符項目數。

使用其中一個函數時，您必須指定要進行全文檢索搜尋的基底資料表。與述詞一樣，您可以指定要搜尋資料表中的單一資料行、資料行清單或所有資料行，而且可以選擇性地指定給定全文檢索查詢將使用之資源的語言。

CONTAINSTABLE 適用的比對種類與 CONTAINS 相同，而 FREETEXTTABLE 適用的比對種類則與 FREETEXT 相同。執行使用 CONTAINSTABLE 和 FREETEXTTABLE 函數的查詢時，您必須明確聯結所傳回的資料列與 SQL Server 基底資料表中的資料列。

5-14 範例 3 -使用 CONTAINSTABLE

下列範例將傳回所有食物種類的描述與類別名稱，其中 Description 資料行包含 "sweet and savory" 的單字，近似於 "sauces" 或 "candies"。所有類別目錄名稱是 "Seafood" 的資料列都會被略過。只會傳回等級值大於或等於 2 的資料列。

```
USE Northwind;
GO
```

```
SELECT FT_TBL.Description,
    FT_TBL.CategoryName,
    KEY_TBL.RANK
FROM Categories AS FT_TBL INNER JOIN
    CONTAINSTABLE (Categories, Description,
        '("sweet and savory" NEAR sauces) OR
        ("sweet and savory" NEAR candies)'
    ) AS KEY_TBL
    ON FT_TBL.CategoryID = KEY_TBL.[KEY]
WHERE KEY_TBL.RANK > 2
    AND FT_TBL.CategoryName <> 'Seafood'
ORDER BY KEY_TBL.RANK DESC;
GO
```

5-15 範例 4 -使用 FREETEXTTABLE

下列範例將擴充 FREETEXTTABLE 查詢，以便先傳回最高等級的資料列，並將每個資料列的等級加至選取清單。若要指定查詢，您必須知道 CategoryID 是 Categories 資料表的唯一索引鍵資料行。

```
USE Northwind;
GO
SELECT KEY_TBL.RANK, FT_TBL.Description
FROM Categories AS FT_TBL
      INNER JOIN
      FREETEXTTABLE(Categories, Description,
```

```
                    'How can I make my own beers and ales?') AS KEY_TBL
    ON FT_TBL.CategoryID = KEY_TBL.[KEY]
ORDER BY KEY_TBL.RANK DESC;
GO
```

下面是相同查詢的擴充，它只傳回等級值大於或等於 10 的資料列：

```
USE Northwind;
GO
SELECT KEY_TBL.RANK, FT_TBL.Description
FROM Categories FT_TBL
    INNER JOIN
    FREETEXTTABLE (Categories, Description,
                    'How can I make my own beers and ales?') AS KEY_TBL
    ON FT_TBL.CategoryID = KEY_TBL.[KEY]
WHERE KEY_TBL.RANK >= 10
ORDER BY KEY_TBL.RANK DESC;
GO
```

5-16 使用布林運算子 - AND、OR、AND NOT 在 CONTAINS 和 CONTAINSTABLE 中

CONTAINS 述詞與 CONTAINSTABLE 函數會使用相同的搜尋條件。這兩個項目都支援使用布林運算子 (AND、OR、AND NOT) 來結合許多搜尋詞彙，以便執行邏輯作業。例如，可以使用 AND 來尋找同時包含 "latte" 和 "New

York-style bagel" 的資料列。例如，可以使用 AND NOT 來尋找包含 "bagel" 但不包含 "cream cheese" 的資料列。值得注意的是，相較之下 FREETEXT 和 FREETEXTTABLE 會將布林詞彙視為要搜尋的單字。

5-17 範例 5

下列範例會使用 **AdventureWorks** 資料庫的 ProductDescription 資料表。此查詢會使用 CONTAINS 述詞來搜尋描述識別碼不等於 5，而且描述同時包含 "Aluminum" 與 "spindle" 這兩個單字的描述。搜尋條件會使用 AND 布林運算子。

```
USE AdventureWorks;
GO
SELECT Description
FROM Production.ProductDescription
WHERE ProductDescriptionID <> 5 AND
    CONTAINS(Description, ' Aluminum AND spindle');
GO
```

5-18 全文檢索查詢的效能微調與最佳化

全文檢索查詢的效能會受到硬體資源的影響，例如記憶體、磁碟速度、CPU 速度和電腦架構。除了這些外部條件以外全文檢索查詢使用者應考慮到如何做好全文檢索查詢的效能微調與最佳化。以 SOL Server 2008 為例，以下是有助於增加全文檢索查詢效能的建議事項：

- 使用 ALTER INDEX REORGANIZE 來重組基底資料表的索引。

- 使用 ALTER FULLTEXT CATALOG REORGANIZE 來重新組織全文檢索目錄。在進行效能測試之前，請確定完成此動作，因為執行此陳述式會造成該目錄中之全文檢索的主要合併。
- 將選擇的全文檢索索引鍵資料行限制為小資料行：雖然支援 900 位元組的資料行,但是建議應在全文檢索索引中使用較小的索引鍵資料行(int 和 bigint 有提供最佳效能）
- 使用整數全文檢索索引鍵：可避免與 docid 對應資料表發生聯結，也因此整數全文檢索索引鍵會依據重要性順序改善查詢效能並改善搜耙效能。如果全文檢索索引鍵也是叢集索引鍵，可能會產生額外效能優勢。
- 將多個 CONTAINS 述詞結合為一個 CONTAINS 述詞：在 SQL Server 中可以在 CONTAINS 查詢中指定資料行清單。
- 如果只需要全文檢索索引鍵或等級資訊，則分別使用 CONTAINSTABLE 或 FREETEXTTABLE，而不要使用 CONTAINS 或 FREETEXT。
- 若要限制結果並增加效能，請使用 FREETEXTTABLE 和 CONTAINSTABLE 函數的 top_n_by_rank 參數。top_n_by_rank 可讓您僅重新叫用最相關的叫用。只有當的商務狀況不需要重新叫用所有可能的叫用 (亦即，不需要「全部重新叫用」(Total Recall)) 時，才應該使用這個參數[4]。
- 檢查全文檢索查詢計畫，確定已選擇適當的聯結計畫，必要的話，請使用聯結提示或查詢提示。如果在全文檢索查詢中使用某個參數，該參數的首次值就會決定查詢計畫。可以使用 OPTIMIZE FOR 查詢提示，強制查詢使用您想要的值進行編譯。這有助於達成決定性的查詢計畫和更佳的效能。

[4] 全部重新叫用通常是法律狀況的必要項目，但是其重要性可能低於商務狀況的效能，例如電子商務。

- 如果全文檢索索引包含過多全文檢索索引片段，可能會導致查詢效能大幅降低。若要減少片段的數目，可以使用 ALTER FULLTEXT CATALOG Transact-SQL 陳述式的 REORGANIZE 選項來重新組織全文檢索目錄。這個陳述式基本上會將所有片段合併成較大的單一片段，然後從全文檢索索引中移除所有已過時的項目。
- 在 SQL Server 2008 全文檢索搜尋中，CONTAINSTABLE (AND, OR) 中指定的邏輯運算子可以實作成 SQL 聯結或在全文檢索執行資料流資料表值函數 (STVF) 內部實作。一般而言，只有一種邏輯運算子類型的查詢是完全由全文檢索執行實作，而混合使用邏輯運算子的查詢也會擁有 SQL 聯結。在全文檢索執行 STVF 內部實作邏輯運算子會使用一些特殊索引屬性，讓它的速度比 SQL 聯結更快。因此，SQL Server 2008 建議您盡可能只使用單一邏輯運算子類型來設計查詢。
- 若為包含選擇性關聯述詞的應用程式，當使用選擇性關聯式述詞與非選擇性全文檢索述詞的查詢撰寫成使用查詢最佳化工具時，這些查詢可能會具有最佳效能。這樣做可讓查詢最佳化工具決定它是否可利用述詞或範圍下推來產生有效的查詢計畫。這種方法比較簡單，而且通常會比將關聯式資料當做全文檢索資料進行索引更有效率。

6. SQL Server 全文檢索服務相關實作

XML 一直是微軟所領導的技術，XML 具備易解讀、標準開放、可延伸、跨平台的優點。一般企業級的系統與應用程式大部份也都與 XML 以及 Web Service 整合。XML 以垂直、階層、遞迴方式描述資料是很適合應用於全文檢索機制中。以下將介紹 SQL Server 2008 全文檢索服務相關實作。

6-1 建立全文檢索目錄

a. 步驟 （以精靈方式產生）：

1. 在 [物件總管] 中，展開伺服器，並展開 [資料庫]，然後展開您要在其中建立全文檢索目錄的資料庫。
2. 展開 [儲存體]，然後以滑鼠右鍵按一下 [全文檢索目錄]。
3. 選取 [新增全文檢索目錄]。
4. 在 [新增全文檢索目錄] 對話方塊中，為您要重新建立的目錄指定資訊。
5. 按一下 **[確定]**。

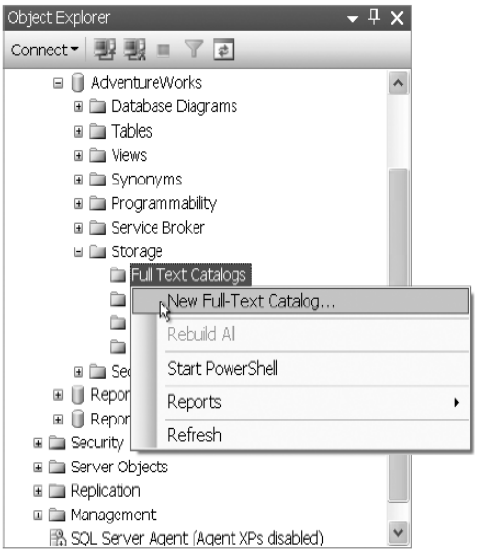


圖 3　建立全文檢索目錄-全文內文清單選項

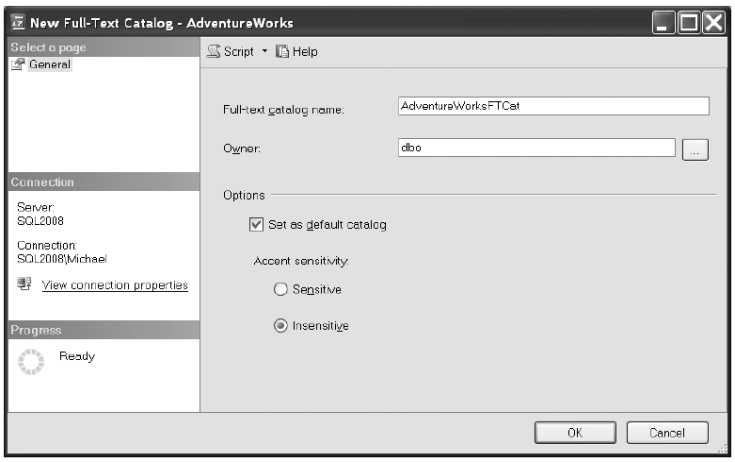


圖 4　建立全文檢索目錄-全文目錄視窗

b. 使用 T-SQL 敘述（直接由指令產生）：

CREATE FULLTEXT CATALOG AdventureWorksFTCat

WITH ACCENT_SENSITIVITY = OFF

AS DEFAULT

AUTHORIZATION dbo;

6-2 建立全文索引

a. 步驟 （以精靈方式產生）：

在 SQL Server 2008 中，可以使用 Management Studio 中的全文檢索索引精靈來建立全文檢索索引。

1. 在 [物件總管] 中，以滑鼠右鍵按一下您要建立全文檢索索引的資料表，然後選取 [全文檢索索引]。

2. 選取 [定義全文檢索索引]。

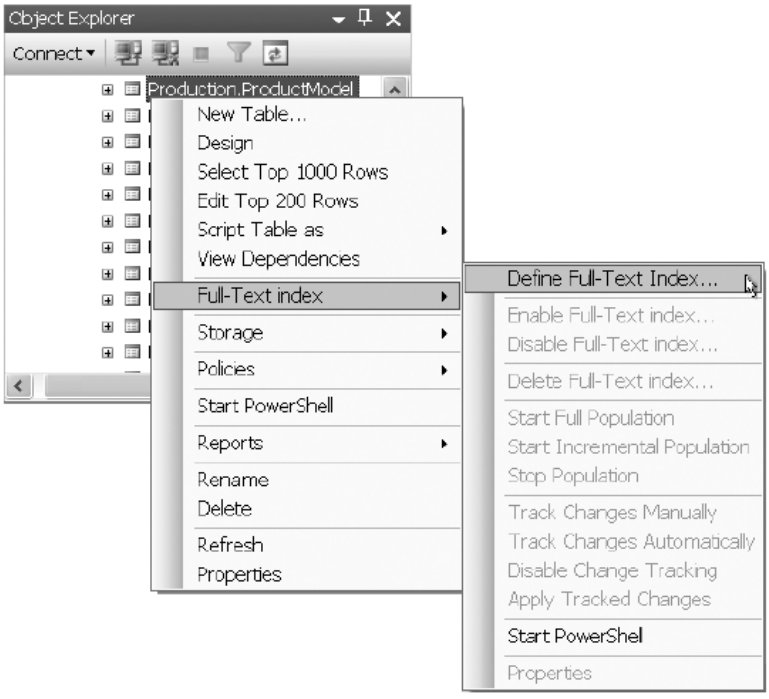


圖 5 建立全文索引-全文索引文件選單

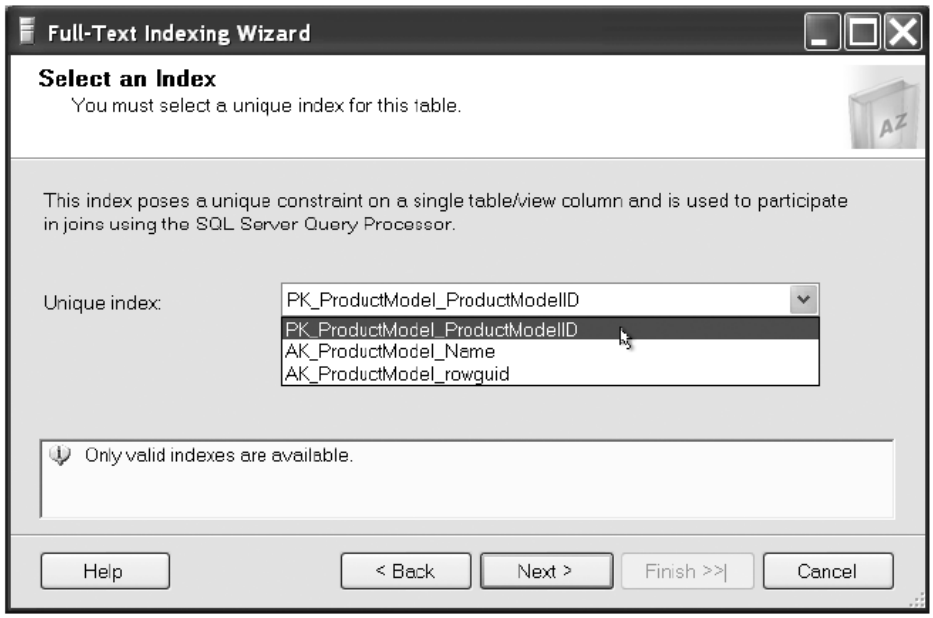


圖 6　建立全文索引-選擇單一欄位唯一索引（unique index）

圖 7　建立全文索引-選擇做為全文索引的欄位與定義 Schema

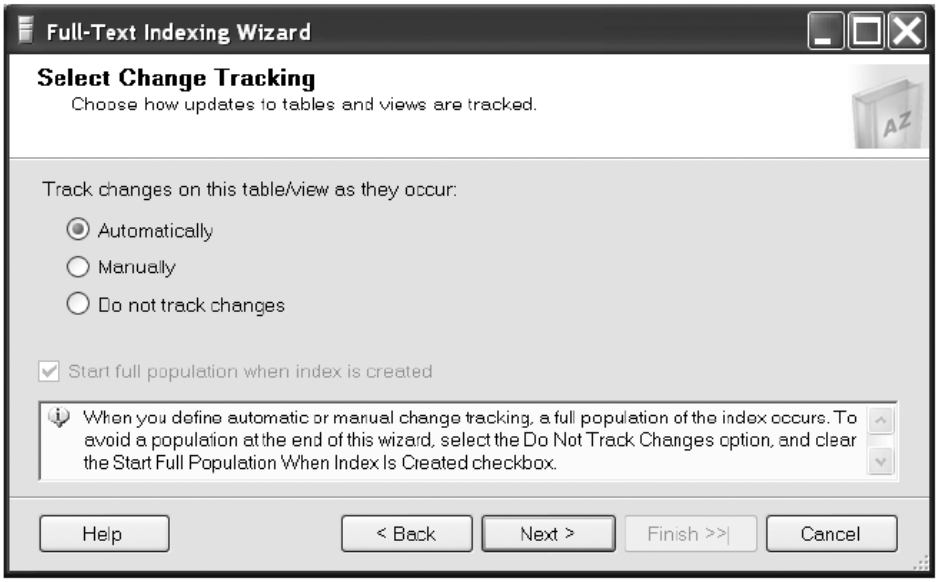


圖 8　建立全文索引-選擇 SQL Server 是否維護全文索引的 log

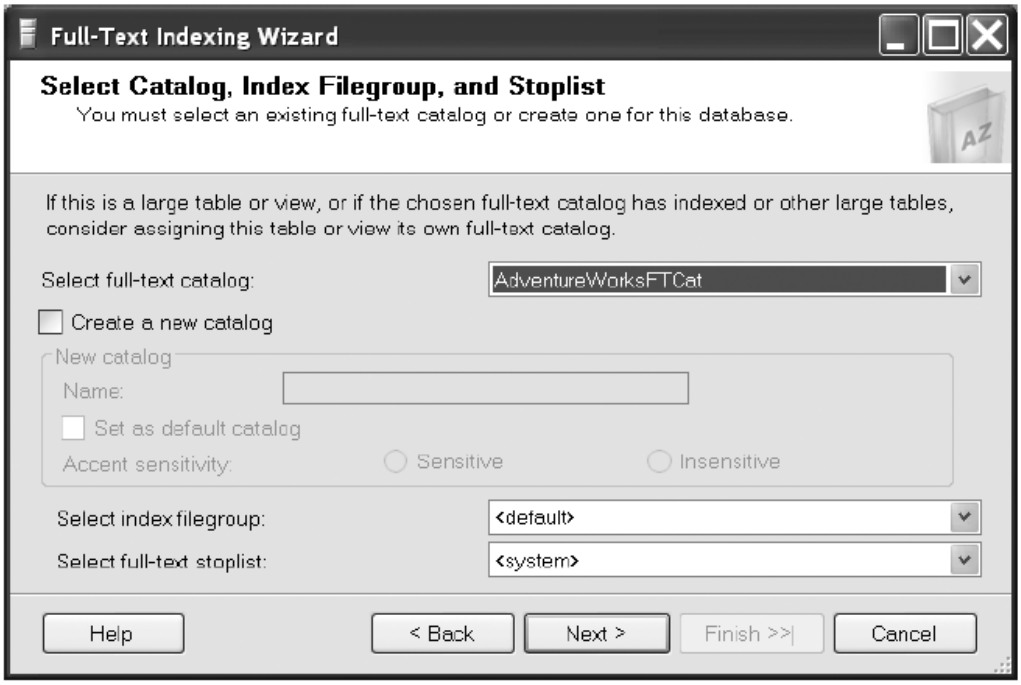


圖 9　建立全文索引-指派全文索引到全文目錄

圖 10　建立全文索引-檢視先前全文索引精靈設定

b. 使用 T-SQL 敘述（直接由指令產生）：

```
CREATE FULLTEXT INDEX
ON Production.ProductModel
(
CatalogDescription LANGUAGE English,
Instructions LANGUAGE English,
Name LANGUAGE English
)
KEY INDEX PK_ProductModel_ProductModelID
ON
(
AdventureWorksFTCat
)
```

```
WITH
(
CHANGE_TRACKING AUTO
);
GO
ALTER FULLTEXT INDEX
ON Production.ProductModel ENABLE;
GO
```

6-3 全文索引查詢

a. FREETEXT -以簡單字元（character-based）為基礎方式

語法：

```
SELECT
ProductModelID,
Name,
CatalogDescription,
Instructions
FROM Production.ProductModel
WHERE FREETEXT(*, N'sock');
```

結果：

Results | Messages

	ProductModelID	Name	CatalogDescription	Instructions
1	18	Mountain Bike Socks	NULL	NULL
2	24	Racing Socks	NULL	NULL

圖 11 全文索引查詢 (FREETEXT) -找尋「sock」之範例結果

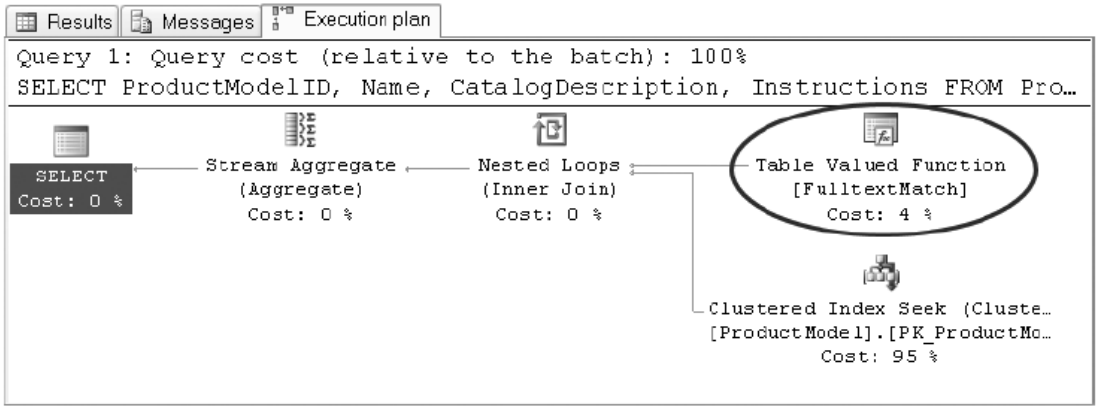


圖 12　全文索引查詢 (FREETEXT) -查詢執行程序

b. FREETEXT -以自動字幹 (*Automatic Word Stemming*) 查詢

語法：

```
SELECT
ProductModelID,
Name,
CatalogDescription,
Instructions
FROM Production.ProductModel
WHERE FREETEXT(CatalogDescription, N'weld', LANGUAGE 1033);
```

結果：

Results | Messages

	ProductModelID	Name	CatalogDescription	Instructions
1	7	HL Touring Frame	NULL	<root xmlns="http://sche...
2	10	LL Touring Frame	NULL	<root xmlns="http://sche...
3	19	Mountain-100	<?xml-stylesheet href="...	NULL
4	25	Road-150	<?xml-stylesheet href="...	NULL
5	47	LL Touring Handlebars	NULL	<root xmlns="http://sche...
6	48	HL Touring Handlebars	NULL	<root xmlns="http://sche...

圖 13　全文索引查詢 (FREETEXT) -查詢「weld」結果 1

ProductMode...	Name	CatalogDescription	Instructions
7	HL Touring Frame	NULL	...Work Center 20 - Frame Welding...
10	LL Touring Frame	NULL	...Work Center 20 - Frame Welding...
19	Mountain-100	...The heat-treated welded...	NULL
25	Road-150	...it is welded and...	NULL
47	LL Touring Handlebars	NULL	...Work Center 20 - Frame Welding...
48	HL Touring Handlebars	NULL	...Work Center 20 - Frame Welding...

圖 14　全文索引查詢 (FREETEXT) -查詢「weld」結果 2

c. CONTAINS -以字詞（word-based）為基礎方式

語法：

```
SELECT
ProductModelID,
Name,
CatalogDescription,
Instructions
FROM Production.ProductModel
WHERE CONTAINS(*, N'weld');
```

結果：

Results | Messages

	ProductModelID	Name	CatalogDescription	Instructions
1	7	HL Touring Frame	NULL	<root xmlns="http://sc...
2	10	LL Touring Frame	NULL	<root xmlns="http://sc...
3	47	LL Touring Handlebars	NULL	<root xmlns="http://sc...
4	48	HL Touring Handlebars	NULL	<root xmlns="http://sc...

圖 15　全文索引查詢 (CONTAINS) -查詢「weld」結果

d. CONTAINS -搭配 *FORMSOF* 產生 term

語法：

```
SELECT
ProductModelID,
Name,
CatalogDescription
FROM Production.ProductModel
WHERE CONTAINS
(
(
Name,
CatalogDescription
),
N'FORMSOF(INFLECTIONAL, sport)'
);
```

結果：

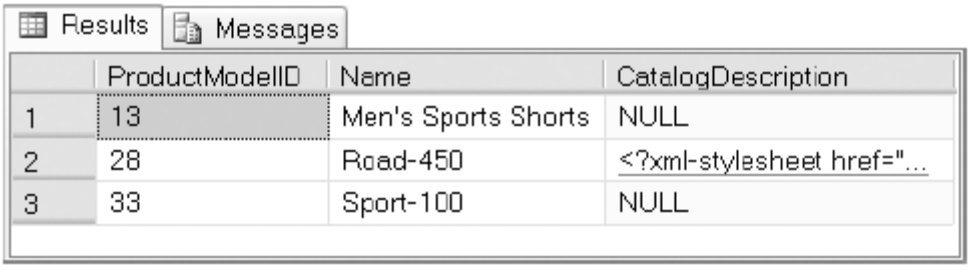


圖 16　全文索引查詢 (CONTAINS) - 搭配 FORMSOF 產生 term

6-4　XML 與全文檢索

a.　使用 GUI 定義 XML

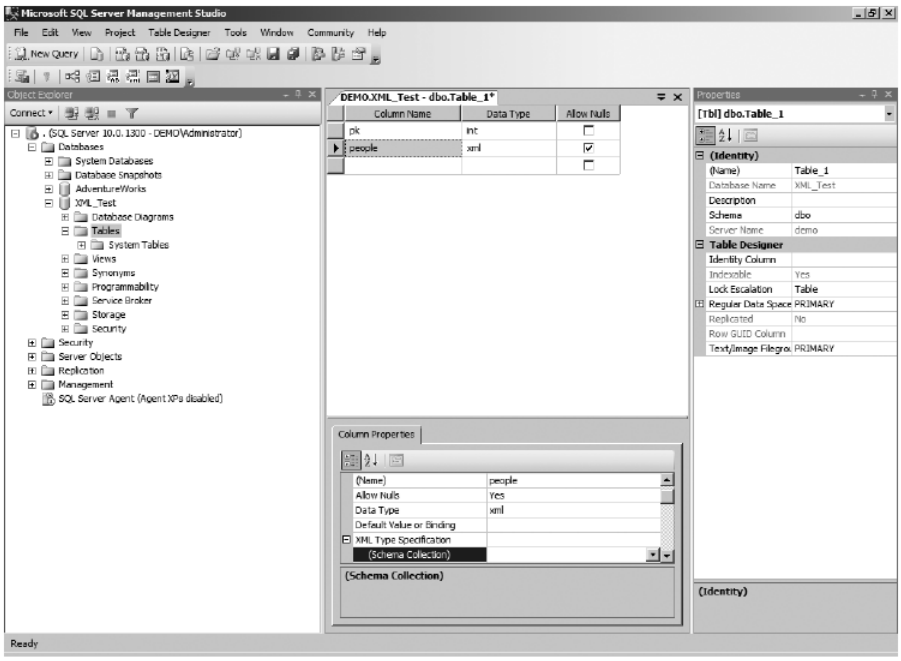


圖 17　XML 與全文檢索- 使用 GUI 定義 XML

b.　輸入資料

使用圖形化介面，設定資料來源於 XML。

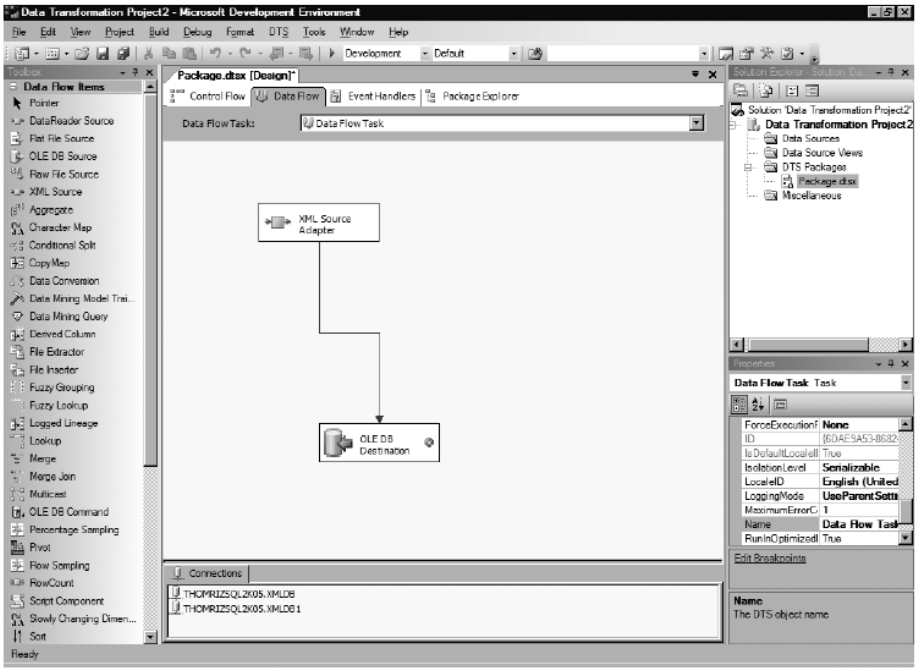


圖 17　XML 與全文檢索- 輸入資料到 XML 欄位

c. 使用 XQuery

XQuery 是可查詢結構化或半結構化 XML 資料的語言。利用 Database Engine 提供的 xml 資料類型支援，就可以在資料庫中儲存文件，然後使用 XQuery 進行查詢。XQuery 是以現有的 XPath 查詢語言為基礎，加上額外支援以獲取更佳的反覆運算、更好的排序結果，以及建構必要 XML 的能力。XQuery 可在 XQuery 資料模型上運作。

建立一個新表格：

```
CREATE TABLE xmltblnew (pk INT IDENTITY PRIMARY KEY, people XML)
GO
--Insert data into the new table
INSERT INTO xmltblnew (people)
SELECT *
FROM OPENROWSET (
BULK 'C:\peopleXML.xml',
SINGLE_BLOB) AS TEMP
GO
```

以年紀查詢所有人，回傳姓名：

```
SELECT people.query(
'for $p in //people
WHERE $p//age
return
<person>
<name>{$p//givenName}</name>
</person>
'
)
FROM xmltblnew
```

7. 感想

a. from Paul

「這是我第二次上這門課，兩次相差時間近十年，對我而言，此次的投入、收穫與經驗完全不同，後者多於前者甚多…」

我第一次修 DB 是在 1999 年第一屆網大碩士學分班修讀黃老師的這門課程，使用的是英文版教科書（Fundamentals of Databade System, Tird Edition, Elmasri & Navathe），以及黃老師自己所編的中文講義（很高興已經有第四版的教科書了，實在值得翻成英文版分享給全世界更多喜好 DB 的人）。與這次相同的是一樣是在網大上聽完老師上課內容的錄音檔,不同的是沒有像實體教室的上課,每次黃老師可以再為我們複習與補充並可以進行實際習題演練互相觀摩學習，在學習的經驗與收穫上截然不同。

在投入上最明顯的不同就是時間的付出，這學期幾乎一星期天天要與 DB 為伍，主要是因為年紀大了衝不快，加上有家庭責任與博班課業上的壓力，不過我依舊採取每天循序前進不鬆懈的方式來學習（我常告訴唐筠這是我的烏龜理論）。當週星期二晚上我會把下週進度看完一次（通常是模糊的尤其在查詢成本的計算上），星期三完成聽完老師的錄音檔（有點概念了），星期四、五開始解習題（常常思考好幾天），星期日的晚上再把習題解完，不懂的地方再閱讀課本與再聽一次錄音檔。我一星期需要花這麼多天與多次的接觸，主要是因為自己的無法有短期吸收與強記很多知識的能力，只能靠反覆與多次的接觸、反思才能將知識變為自己的經驗。雖然如此投入，不過我還是無法應付在有限時間下的考試型態，所以考試成績還是很不理想。DB 是我這學期投入最多時間的課程，很辛苦，但很慶幸的是我依舊很用力與很認真的把它學完，即使自己的年紀或其他外在因素的壓力下。不管如何，我實在是很幸運，我既是畢業於中山資管碩班的學生，若沒有能親自面對面上到黃老師的課，那對我一定是一項很巨大的損失。

關於期末專題的部份，我與唐筠選擇的是「Full Text Index Searching」。在業界我於 1995 接觸到 IBM Lotus Notes Domino 系統，是當代唯一一套可以全文檢索的系統，當時我委外給 IBM 公司以 Lotus Notes Domino 技術開發一套飛機技令全文檢索系統。為何會選擇它是因為當時的 Oracle 等相關 DBMS 等並不支援全文檢索技術，甚至遇到當使用者提出應用系統絕大部分的輸入欄位或查詢欄位是無法估算欄位長度與不一定可以有哪些固定欄位可以做全文檢索時，是一件非常棘手的 IS 開發問題。我在當時就對全文檢索這項技術充滿了好奇，不過由於後來已漸漸走向高階管理工作，對於 IS 技術部份已漸漸生疏，不過很慶幸的事隔 13 年，我還是可以透過黃老師 DB 課程來重新了解新一代的全文檢索技術發展現況，也解了我心中多年之謎。

這次的期末專題我與唐筠合作的很好，也看到了唐筠認真執著的要把每一件工作做到最好的態度，我很欣賞。其時我們在 Due Date 兩星期前大致已經完成了這次的 Project 報告，不過我們還是繼續對這份 Project 再次的檢視與加強，希望能夠提供最充足的資訊給未來有機會讀到它的人。最後，真的深深地謝謝黃老師讓我從您身上學習到認真與務實的幫榜樣。謝謝黃老師，智義 敬上。

b. from Tiffany

「這是我距離從校園中離開再重新回校園，相距十年的時間，感覺很充實且有份量的課程。對我而言，所得到的收穫除了知識與技術上更深入的了解，也實務應用到工作，受益良多。而無法以量化指標論斷的是從老師認真的教學態度上，學到了更多對於追求知識、做研究應保有的態度」

學期近了尾聲，回顧這一個學期自我投入課程與練習的時間 - 時多時少。開學時拿到課程大綱，每周課堂前要預習、要看老師的數位影音教材、還要記得練習課本的習題，課堂中，老師以深入該週議題相關的內容做為補充語說明，並

且以互動討論的方式進行，且進行習題報告與討論。如此，每週有很充實的課程份量，但是，如果有一週沒有按照既定進度進行，後果就是如堆雪球越滾越大，；有時候就會形成日後要花更多倍的精神與時間閱讀理解。而對於時間管理一直做的不是很妥善的人來說，是一個很大的考驗，因此，相對反應於我不好的成績。但是，一個學期這樣的訓練，的確是有讓我自己漸漸調整與改善時間管理。

期末專案我和智義合作，智義是屬於謹慎用功與認真類型，遇上我是屬於有想法、天馬行空的思考評估可行性與找到可能的框架之後，才開始進行，所以往往在時間的掌控上，都是讓夥伴捏把冷汗的類型。而在全文檢索的議題中，之前念碩士班的時候是針對中文應用，但是，當時確切而言是僅進行到斷詞和與語意分析，在全文檢索的時候，應用的比例較少。當時並沒有像現在這麼多的套裝應用或是資料參考，當然，也還沒有見到中研院那套很有名的系統。經過十年後，能在老師的課程中，再次回頭看以往與中間發展過程到現在的趨勢，對於全文檢索有更多更豐富的知識與了解更廣的應用。不過，在此有一個小小的建議 XML 章節真的很重要，希望這一章節在課程中是可以聆聽老師的教授與指導。

非常謝謝老師這學期對於我這不是很認真的學生的包容，但是，經由這一門課程的學習，讓我除了學到更身的資料庫系統相關知識，也漸漸在老師謹慎認真教學的潛移默化的精神中，融合到自己的工作與生活，謝謝老師～也祝福老師寒假快樂。

參考文獻

英文：

[1] **Pro SQL Server 2008 XML**, ch7- Indexing XML, p177-192 ,Apress , 2008

[2] **Accelerated SQL Server ,** ch17 - SQL Server XML and XQuery Support **,** p515-552**,**Apress, 2008**,**

http://www.springerlink.com/content/j704158050642l96/fulltext.pdf

[3] **SQL Server 2008 Transact-SQL Recipes ,** ch6 - Full-Text Search, Apress , p217-237, 2008

http://www.springerlink.com/content/vql7452131w6815k/fulltext.pdf

[4] **Pro T-SQL 2008 Programmer's Guide**, ch10 - Integrated Full-Text Search Apress, p273-298, 2008

http://www.springerlink.com/content/k12u8x8348014xx4/fulltext.pdf

中文：

[胡述兆, 1995] 胡述兆，圖書館學與資訊科學大辭典，國立編譯館主編，台北市: 漢美圖書，1995，頁 669。

[林東清, 2008] 林東清著，資訊管理-- e 化企業的核心競爭能力，台北市:智勝文化，2008，頁 281 - 327。

[中文全文檢索搜索網, 2004] 中文搜索引擎技術揭密：系統架構，

http://www.FullSearcher.Com/n200491114034735.asp

[徐玉梅, 2002] 研究文獻全文資料庫建置與檢索之規劃研究，行政院國家科學委員會九十一年度科技行政研究發展報告，行政院國家科學委

員會科學技術資料中心，

http://web1.nsc.gov.tw/public/Data/85718153271.pdf

網站：

[1] 全文檢索搜尋 (SQLServer) ,MSDN 開發技術,

http://msdn.microsoft.com/zh-tw/library/ms142571.aspx

[2]微軟技術網站, http://technet.microsoft.com/zh-tw/library/aa337082.aspx

[3]IBM Open Find,

http://www.demos.ibm.com/on_demand/Demo/zh_tw/IBM_Demo_IBM_OmniFind_Yahoo_Edition-Dec06.html

[4] Lucene, http://incubator.apache.org/lucene.net/

[5] Google, http://www.google.com.tw/

[6] Oracle , http://www.oracle.com/global/tw/index.html

[7] MySQL, http://www.mysql.com/

[8] Wiki 「全文檢索」議題,

http://zh.wikipedia.org/w/index.php?title=%E5%85%A8%E6%96%87%E6%AA%A2%E7%B4%A2&variant=zh-tw