第四章 資訊檢索技術

鄭卜壬、李家豪

資訊檢索是擷取、組織和利用資訊的重要技術領域。隨著數位典藏資料急遽成長，若使用者無法得知有哪些典藏資料符合其需求，再多的典藏內容對其而言，也缺乏使用價值。本章將探討與數位典藏相關之資訊檢索技術，依資料型態的不同，資訊檢索技術大致可分為三大類：資料庫檢索、文件資訊檢索與多媒體資訊檢索（註：有時資訊檢索是專指文件及多媒體資訊檢索，而資料庫檢索稱為資料檢索(Data Retrieval)。

「資料庫檢索」主要探討結構化資料的檢索方式，其應用範圍以傳統資料庫為主，例如關聯式資料庫(Relational database)利用表格式的結構化資料模型，儲存文物典藏品的各種屬性，並提供許多檢索功能，以協助使用者快速搜尋符合某些欄位條件的典藏記錄。

「文件資訊檢索」主要探討非結構化文件資料的檢索方式，其應用範圍以搜尋引擎為主，例如一般搜尋引擎提供關鍵詞的查詢介面，協助使用者快速檢索非結構化的網路文件。

「多媒體資訊檢索」主要探討圖像與影音等其他媒體型態資料的檢索方式，由於圖像與影音具有獨特的媒體特性，使用者可能有不同於前兩種技術的檢索需求，例如尋找相似圖像或曾聽過的新聞影音片段。

本章著重於介紹資料庫與文件資訊檢索技術，多媒體資訊檢索與管理的相關介紹與說明，請參考第六章「影音管理」。

第一節　資料庫檢索

隨著網路基礎建設之迅速普及，傳統靜態網頁內容設計已經不敷所需。目前許多數位典藏系統利用資料庫儲存資料，依不同需求設計各種動態網站。由於一般資料庫皆提供檢索功能，不需額外建置搜尋引擎，即可輕易使典藏網站具備搜尋能力。依資料模型的不同，目前存在多種類型的資料庫，例如：結構化的關聯式資料庫、半結構化的 XML 資料庫及物件導向式資料庫(Object-oriented Database)等。在商業應用上，仍以較成熟且常用的關聯式資料庫為主，所以本節內容將以關聯式資料庫的基本概念為主，並著重於如何提升其搜尋速度。

一、關聯式資料模型

簡而言之，資料模型是用來規範如何紀錄資料的規則與架構。資料模型的相關理論不少，目前最著名也最常用的資料模型為「關聯式資料模型」(Relational Data Model)，最早由 E. F. Codd 於 1970 年(Codd, 1970)在「A Relational Model of Data for Large Shared Data Banks」這篇論文中提出。關聯式資料模型，以數學集合為理論基礎，所有資料均紀錄於二維表格中，表格與表格間可以透過多種運算模式，查詢所需要的資料，其中包含四種傳統集合基本運算：聯集（Union）、交集（Intersection）、差集（Difference）與乘積（或稱卡式積，Cartesian product），以及四種關聯代數基本運算：選取（Selection）、投影（Projection）、合併（Join）與除法（Division）。經過運算，結果可得到新的關聯式表格（Relation in, relation out），使關聯式資料模型能夠處理許多複雜的查詢需求。

如下表 4- 1 所示，表格（Table）可分為列（Row）與行（Column），其中列亦稱為記錄（Record 或 Tuple），行亦稱為屬性（Field 或 Attribute）。

表 4- 1：拓片基本資料表

編號(ID)、分類(DB_TYPE)及主題(TITLE)

Column / Field / Attribute

ID	DB_TYPE	TITLE
1	漢代石刻畫像	武氏北面畫像
2	甲骨文	甲骨刻辭
3	青銅器	洹子孟姜壺
4	佛教石刻造象	劉根四十一人浮圖

Row / Record / Tuple

表 4- 2：拓片編號及主題

ID	TITLE
1	武氏北面畫像
2	甲骨刻辭
3	洹子孟姜壺

表 4- 3：拓片編號及主題

ID	TITLE
1	武氏北面畫像
3	洹子孟姜壺
4	劉根四十一人浮圖

表 4- 4：拓片分類之相關書籍

DB_TYPE	BOOK
甲骨文	殷墟甲骨刻辭類纂
甲骨文	甲骨文字詁林
青銅器	商周青銅紋飾

以下將利用上述四個表格（表 4- 1 到表 4- 4）分別介紹關聯式資料模型的 8 個基本運算。請注意這些表格只是為了方便舉例與說明，在實務上，讀者可應用其他語意模型（如 ER 模型 (Chen, 1976)），在表格的設計上，建議利用表格正規化(Normalization)的方式來建立表格內容架構，與維護表格間的關聯性，細節可參考資料庫系統的相關書籍介紹，如「An introduction to database systems」(Date, 2003)及「Database systems concepts」(Silberschatz, Korth, & Sudarshan, 2005)。

若每個表格可視為一個集合，則表格中每一列表示該集合中的一個元素(Element)，依此原則，可以很容易在表格上進行傳統集合運算，例如：

（一）聯集（以∪表示）

表 4- 5：表 4- 2　∪表 4- 3（包含兩表格內所有的記錄）

ID	TITLE
1	武氏北面畫像
2	甲骨刻辭
3	洹子孟姜壺
4	劉根四十一人浮圖

（二）交集（以∩表示）

表 4- 6：表 4- 2 ∩ 表 4- 3（包含兩表格內共有的記錄）

ID	TITLE
1	武氏北面畫像
3	洹子孟姜壺

（三）差集（以－表示）

表 4-7：表 4-2　－表 4-3（包含表 4-2 而不在表 4-3 中的記錄）

ID	TITLE
2	甲骨刻辭

（四）乘積（以 × 表示）

表 4-8：表 4-2　×　表 4-4（包含表 4-2 及表 4-4 所有記錄的序對集合）

ID	TITLE	DB_TYPE	BOOK
1	武氏北面畫像	甲骨文	殷墟甲骨刻辭類纂
1	武氏北面畫像	甲骨文	甲骨文字詁林
1	武氏北面畫像	青銅器	商周青銅紋飾
2	甲骨刻辭	甲骨文	殷墟甲骨刻辭類纂
2	甲骨刻辭	甲骨文	甲骨文字詁林
2	甲骨刻辭	青銅器	商周青銅紋飾
3	洹子孟姜壺	甲骨文	殷墟甲骨刻辭類纂
3	洹子孟姜壺	甲骨文	甲骨文字詁林
3	洹子孟姜壺	青銅器	商周青銅紋飾

除了集合運算，以下四種關聯代數運算有助於查詢表格中的資料，包含選取、投影、合併與除法。

（一）選取（以 α 表示）：α_C(Table)

選取運算可以從表格中選出符合條件 C 的記錄（列）。例如：$\alpha_{ID<2}$（表 4-92）表示從表 4-2 中選出 ID 小於 2 的記錄。

表 4-9：從表 4-2 中選出 ID 小於 2 的記錄

ID	TITLE
1	武氏北面畫像

（二）投影（以 π 表示）：$\pi_{Field\ 1, Field\ 2, ..., Field\ n}$(Table)

投影運算可以從表格中選出一個或多個(Field 1, …, Field n)欄位（行）。例如：π_{TITLE}（表 4-102）表示從表 4-2 中選出 TITLE 欄位。

表 4- 10：從表 4- 2 中選出 TITLE 欄位

TITLE
武氏北面畫像
甲骨刻辭
洹子孟姜壺

（三）合併（以 ⋈ 表示）：Table ⋈ Table

合併運算又可細分為多種運算方式，在此介紹自然合併(Natural join)運算。自然合併運算用以結合兩個表格，其考慮兩個表格內的每一筆記錄，假設記錄 1 來自於一個表格，記錄 2 來自於另一個表格，若此 2 個記錄有共同的欄位，且該欄位內的值相同，則自然合併運算會將記錄 1 與記錄 2，加入合併後的新表格。例如：表 4- 1 ⋈ 表 4- 4，表示依據 DB_TYPE 欄位合併表 4- 1 及表 4- 4。

表 4- 11：依據 DB_TYPE 欄位合併表 4- 1 及表 4- 4 之結果

ID	DB_TYPE	TITLE	BOOK
2	甲骨文	甲骨刻辭	殷墟甲骨刻辭類纂
2	甲骨文	甲骨刻辭	甲骨文字詁林
3	青銅器	洹子孟姜壺	商周青銅紋飾

（四）除法（以 ÷ 表示）

除法可以從一表格中選出包含另一表格中所有記錄的記錄。例如，當表 4- 12 為如下內容時：

表 4- 12：除法例示

BOOK
殷墟甲骨刻辭類纂
甲骨文字詁林

則表 4- 4÷表 4- 12，表示有哪些表 4- 4 的 DB_TYPE 包含表 4- 12。表 4-13

表 4- 13：表 4- 4 ÷ 表 4- 12 所得結果。

DB_TYPE
甲骨文

值得注意的是，上述基本運算的結果亦為一個表格，換言之，此結果允許使用者同時結合多種運算，以進行複雜的查詢。例如：假設表 4- 1 記錄各個拓片的編號(ID)、分類(DB_TYPE)及主題(TITLE)，表 4- 4 記錄各種拓片分類(DB_TYPE)

及相關書籍(BOOK)，若要利用表 4- 1 和表 4- 4 查詢「甲骨刻辭」可能相關的書籍，可結合選取、合併與投影三種運算如下：

π_{BOOK} （ $\alpha_{TITLE=甲骨刻辭}$ （ 表 4- 1 $\bowtie$ 表 4- 4 ） ），

讀者可嘗試由上述合併運算之範例結果表格（參考表 4- 11），再利用選取與投影運算方法，得到「殷墟甲骨刻辭類纂」與「甲骨文字詁林」的結果，當然在此只是舉一種可能的運算方式為例，不同的結合方式會有不同的運算複雜度。

二、SQL 結構化查詢語言

在關聯式資料模型中，重要的檢索議題之一，是如何有效地以 SQL 結構化查詢語言快速找到所需資料，本節簡單介紹 SQL 的主要語法，以及最佳化 SQL 查詢的方法。

（一）何謂結構化查詢語言（Structured Query Language，簡稱 SQL）

結構化查詢語言為一種文字敘述語言，藉由該語言可描述使用者的查詢需求，進而從資料庫中取出滿足條件的資料。目前 SQL 已成為使用關聯式資料庫的標準語言，由美國國家標準協會（ANSI）與國際標準組織採用為國際標準，美國國家標準協會於 1992 年推出 SQL-92 (ISO/IEC JTC1/SC21, 1992)，是目前資料庫廠商主要支援的標準，各資料庫廠商可依據其特殊需求再自行擴充。例如，Oracle 自行擴充的 SQL 為 PL/SQL，而 Microsoft SQL Server 自行擴充的 SQL 則命名為 Transact-SQL，也因此，不同資料庫間可能會有 SQL 敘述彼此不相容的情況發生。

SQL 依照用途可簡單區分為三大類：資料定義語言（Data Definition Language ，簡稱 DDL）、資料操作語言（Data Manipulation Language，簡稱 DML）及資料控制語言（Data Control Language，簡稱 DCL），分別介紹如下：

1. 資料定義語言（DDL）

資料定義語言是用來建立、變更或刪除資料庫內物件（例如：Table、View、Trigger、Index 等）的 SQL 敘述。可分為下列 SQL 敘述：

(1)CREATE 建立物件
(2)ALERT 變更物件
(3)DROP 刪除物件

2. 資料操作語言（DML）

資料操作語言（DML）是用來搜尋、新增、刪除與修改資料的 SQL 敘述。

可分為下列 SQL 敘述：

(1)SELECT 搜尋資料
(2)INSERT　新增資料
(3)DELETE　刪除資料
(4)UPDATE 更新資料

3. 資料控制語言（DCL）

為達到資料庫中資料正確與資料安全的目的，資料庫提供存取權限控管（Access Permission）與交易流程控管（Transaction）的相關 SQL 敘述，即為資料控制語言（DCL），包含下列 SQL 敘述：

(1)GRANT　授權
(2)REVOKE　撤回權限
(3)COMMIT　交易確定
(4)ROLLBACK　交易取消

在資料庫檢索方面，SQL 利用「Select-From-Where」的方式查詢表格資料，此方式可描述使用者想要從哪些表格中(From)找尋符合特定條件的資料(Where)，並在結果表格中選出想要看的欄位(Select)。以本章第一節之一中的表 4- 1 為例，若使用者想在表 4- 1 中查詢「甲骨刻辭」的標號(ID)及分類(DB_TYPE)，其 SQL 查詢語言可表示成：


```
Select ID, DB_TYPE
From 表 4- 1
Where 表 4- 1.TITLE=甲骨刻辭
```

其中 From 敘述表示使用者欲查詢表 4- 1，Where 敘述表示使用者欲查詢表 4- 1 中所有 TITLE 為「甲骨刻辭」的記錄，其中「表 4- 1.TITLE」是指表 4- 1 的 TITLE 欄位，最後 Select 敘述表示輸出這些記錄的 ID 及 DB_TYPE，亦即「甲骨刻辭」的標號及分類。

若使用者想查詢「甲骨刻辭」可能相關的書籍，由於表 4- 1 中缺乏書籍資料，需再配合表 4- 4 的內容，使用者可合併表 4- 1 及表 4- 4 以進行查詢：

```
Select T4.BOOK
From 表 4- 1 T1, 表 4- 4 T4
Where T1.DB_TYPE=T4.DB_TYPE And T1.TITLE=甲骨刻辭
```

其中 From 敘述表示使用者欲查詢的表格為表 4- 1 和表 4- 4，並將表 4- 1 命

名為 T1、表 4- 4 命名為 T4，Where 敘述表示使用者欲查詢的記錄需符合的條件，即表 4- 1 和表 4- 4 中記錄的 DB_TYPE 值需相同（依此條件結合表 4- 1 和表 4- 4 的記錄，可得到本章第一節之一中合併運算範例的結果，即表 4- 11，並且，使用者只關心合併表格中 TITLE 為甲骨刻辭的記錄，最後 Select 敘述表示，在上述合併後且符合 TITLE 條件的記錄中，使用者只想知道 BOOK 欄位的資料，亦即「甲骨刻辭」可能相關的書籍。由上述二例顯示，使用者可藉由簡單的 Select-From-Where 方式表達查詢語意，告訴資料庫系統使用者的查詢需求。相對地，就關聯式資料庫系統而言，其可利用上節所述之資料模型中提供的基本運算，把查詢語言轉換成一連串電腦可執行的基本運算動作，以搜尋使用者想要查詢的資料範圍。

由於本書篇幅關係，本章節並不詳細介紹 SQL 用法，上述只是簡單將 SQL 做分類、列出數項 SQL 語法以供區別與參考，讀者可參考關聯式資料庫之相關書籍(Date, 2003)(Silberschatz, Korth, & Sudarshan, 2005)，快速了解基本的 SQL 語法與應用。以下將探討 SQL 查詢語言最佳化及資料庫檢索在實作時經常碰到的問題。

三、SQL 最佳化

（一）SQL Optimizer

目前常用關聯式資料庫產品大部分皆具備 SQL Optimizer 功能，Optimizer 會從各層面去剖析使用者所執行之 SQL 語法，最後選擇出最合適的執行計畫（Execution Plan）去執行。因此，在介紹如何最佳化 SQL 前，必須先了解何謂 SQL Optimizer 及其運作流程。

SQL Optimizer 主要決策流程可分為四個步驟：

1. 步驟一：SQL 語法檢查與剖析
   (1)檢查 SQL 語法是否有誤，若有語法上的錯誤則回傳錯誤訊息。
   (2)剖析 SQL 語法，自動進行 SQL 語法轉換，藉以增加語法容錯性。例如：資料型態的自動轉換。

2. 步驟二：選擇欲使用之索引與資料表結合方式
   (1)選擇相關可用之索引與各種資料表結合方式。
   (2)建立各式可能適合之執行計畫，此執行計畫流程則稱為 Access Path。

3. 步驟三：評估各執行計畫所需花費成本
   根據 SQL 所使用到之表格資料量、資料分布狀況、索引狀態等相關統

計資訊，預估各個執行計畫所需花費時間與系統資源。

4. 步驟四：決定執行計畫

成本越低的執行計畫越有效率，因此選取最低成本之執行計畫，交由 SQL 處理器(SQL Processor) 執行。

下圖 4- 1 顯示此四個步驟的流程。

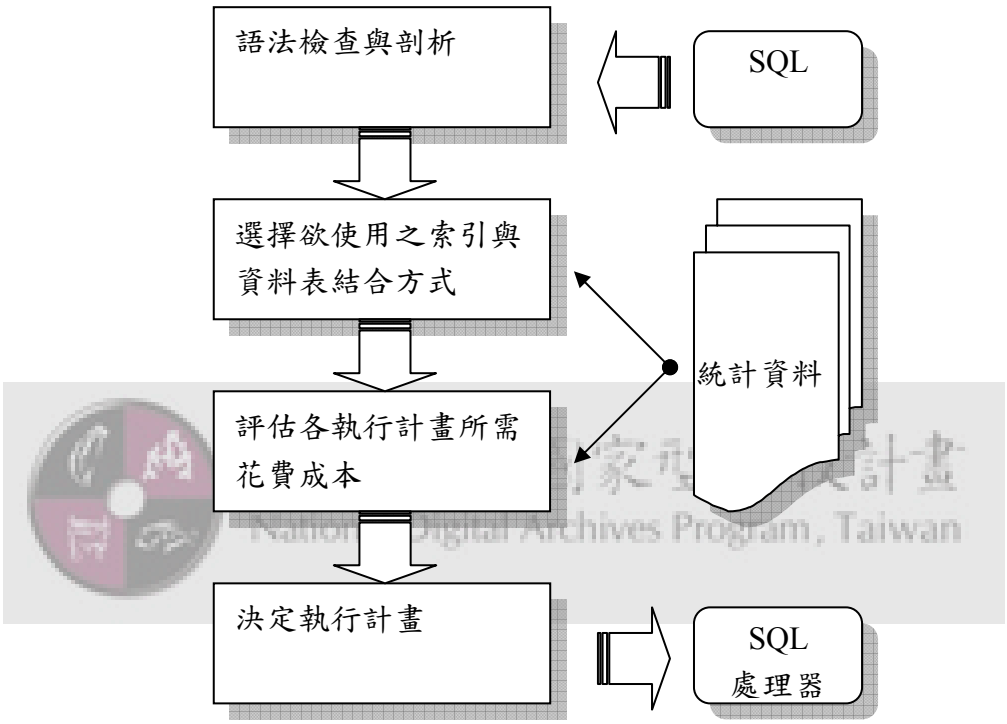


圖 4- 1：SQL Optimizer 決策流程圖

一般而言，SQL Optimizer 在挑選執行計畫時，所採取的方式有 Rule-Based Optimizer（RBO）與 Cost-Based Optimizer（CBO）兩種方式，上面所介紹之「步驟三：評估各執行計畫所需花費成本」為 CBO，也是一般資料庫最常用之評估方式。

1. Rule-Based Optimizer（RBO）

純粹依據 SQL 敘述內容及相關索引來決定執行計畫，換言之，RBO 建立 Access Path 時，並不會考慮到資料表之資料量、資料分布等統計資訊。若有多個執行計畫時，則會依據資料庫內定之優先順序，選取最高之優先權作為最佳執行計畫，例如：索引掃瞄方法(Index Scan)的優先權就會比表格掃瞄方法(Table Scan)之逐筆搜尋來得高。一般認為索引掃瞄方法比表格掃瞄方法有效率，但對小型資料表而言，使用表格掃瞄方法之逐筆搜尋，有可能

比索引掃瞄方法來得快，或者當資料表中，符合搜尋條件之筆數在總筆數中佔較高的比例時，有時使用表格掃瞄方法也會比較快。

2. Cost-Based Optimizer（CBO）

依據統計資料，量化每個 Access Path 之成本，其中成本包含時間與系統資源成本，例如：執行 SQL 所需中央處理器(CPU)、記憶體及硬碟存取時間；換言之，要有統計資料才能使用 CBO，若無統計資料，就無法使用 CBO 方式，只能改用 RBO 評估方式。

當使用 CBO 評估方式時，即使評估相同 SQL，但在不同時間點，因為資料有所異動，導致資料量、資料分布不同，統計出來的資料也不盡相同。Optimizer 可能會採取不同 Access Path 決策，故適時更新統計資料對 CBO 評估方式而言是非常重要，否則可能會讓 Optimizer 選擇不適當的執行計畫。

目前各種資料庫用來取得統計資料的 SQL 語法並不相同，以常見之 Oracle 與 MS SQL Server 為例，Oracle 使用「ANALYZE」，而 MS SQL Server 使用「UPDATE STATISTICS」，例如：

Oracle：

```
ANALYZE {INDEX <index name> | TABLE <table name>}
COMPUTE STATISTICS
```

MS SQL Server：

```
UPDATE STATISTICS <table name> [<index name>]
```


了解 Optimizer 運作的整個流程後，其次需了解當 Optimizer 在建立執行計畫時，資料表間結合方式的選擇，因其影響查詢效能甚鉅。一般而言，若查詢兩個以上資料表時，Optimizer 會將此 SQL 查詢視為一組多重合併（Multi-Join）。

```
Select A.Field, B.Field, C.Field
From A, B, C
Where A.PK = B.PK And B.PK = C.PK
```

例如，當結合 A、B、C 三個資料表時，有時 Optimizer 會先結合前 A、B 兩個資料表，取得結果後，再將 A、B 前兩個資料表結果合併到第三個資料表 C，此時，前兩個資料表結合所得之結果筆數多寡，將會影響與第三資料表結合的查詢效能，對某些資料庫系統，在撰寫 SQL 時，資料表結合順序就顯得相當重要。

而 Optimizer 在選擇資料表結合方式時，一般會有下列三種結合方式：

1. SORT MERGE JOIN

如圖 4- 2，使用 SORT MERGE JOIN 結合兩個資料表時，首先會個別透過表格掃瞄方法，以執行的 SQL 查詢語法為篩選條件，分別找出並排序(Sort)兩個資料表，然後再將這兩個排序過的結果，執行合併(Merge)動作，取得最終查詢結果。所以 SORT MERGE JOIN 主要可分為排序及合併兩個動作，一般情況下排序會比合併耗時，而且合併前必須等兩個資料表都排序完畢後才能進行，若兩個資料表大小不一時，大表格的排序動作就會影響到整體查詢效率，故 SORT MERGE JOIN 方式較適合資料表間大小差異較小時使用。

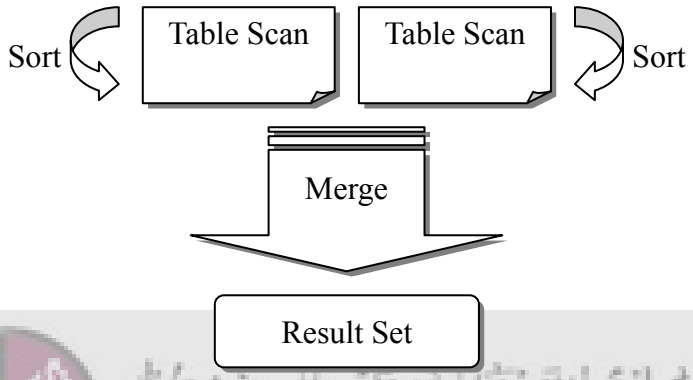


圖 4- 2：SORT MERGE JOIN 的概念

2. NESTED LOOP JOIN

如圖 4- 3，NESTED LOOP JOIN 將欲結合的兩個資料表，分為外部資料表（Nested Parent）與內部資料表（Nested Child）。先採用表格掃瞄方法搜尋外部資料表，然後再使用內部資料表之索引，來查詢內部資料表之所有符合外部資料表的結合條件，取得最後查詢結果。所以當兩個欲結合之表格大小差異頗大，即外部資料表資料量少，內部表格資料量大且有設有索引時，較適合採用 NESTED LOOP JOIN，可加快查詢速度。

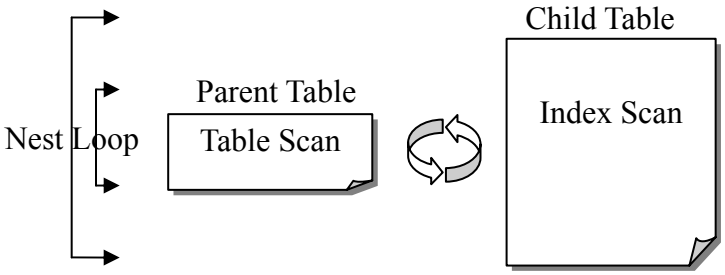


圖 4- 3：NESTED LOOP JOIN 的概念

3. HASH JOIN

如圖 4- 4，HASH JOIN 從欲結合的兩個資料表中，選出資料量較小的資料表為 Build Table，另一較大資料表為 Probe Table，首先用表格掃瞄方法搜尋 Build Table，為其每一筆資料建立雜湊值（Hash Key），並放入雜湊表（Hash Table）中，接著同樣針對較大資料表 Probe Table 進行表格掃瞄，算出每一筆資料雜湊值，最後再根據結合條件，取出雜湊表中有同樣雜湊值的列，取得搜尋結果，此種方式適用於兩個資料表大小差異較大時。

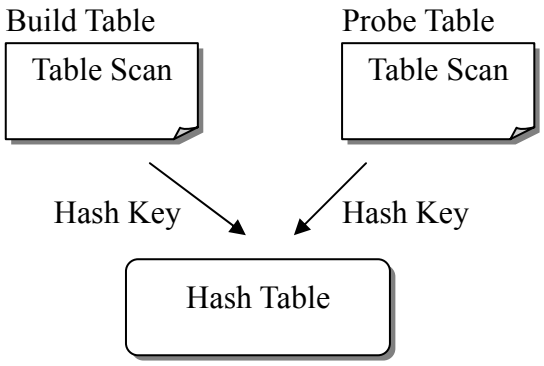


圖 4- 4：HASH JOIN 的概念

（二）如何最佳化 SQL

在了解 Optimizer 的相關運作過程後，接下來介紹如何最佳化 SQL 流程與注意事項：

1. 良好 Database Schema 設計

   何謂 Database Schema？一個資料庫可以有多個 Schema，每個 Schema 可包含多個物件，物件包含 Table、View、Procedure、Index……等，當 SQL 執行於 Schema 的物件時，若有不當 Schema 架構設計，則會使 SQL 執行效率變得遲緩而無效率。

2. 事先確認與收集完整 SQL

   在系統建置初期，負責 Database Schema 規劃人員，應與程式開發人員密切溝通，盡量事先確認與收集完整 SQL，調適出 SQL 最佳撰寫方式，甚至必要時，Schema 也需作適當調整。

3. 建立「合適」的索引

   收集完整 SQL 後，應思考這樣的 SQL 撰寫方式，會讓 Optimizer 如何建立執行計畫？索引要如何建立才會被使用？或者反向去思考 SQL 要如何撰寫才能使用到此索引，必要時再回去修改 SQL 語法，特別要注意搜尋多個資料表的 SQL。此外，索引也不是建越多越好，因

為當原始資料有被異動（新增、修改、刪除）時，索引也會隨之更新，這會增加資料異動時的系統資源使用，也會增加儲存空間。

（三）設計 SQL 之注意事項

以下介紹在數位典藏系統實作中，撰寫 SQL 查詢語言時，常常被忽略的一些事項。因不同資料庫系統會針對不同項目進行最佳化處理，這些注意事項可能並不適用於所有現存之關聯式資料庫，但是其有助於解決在一般情形下可能面臨的問題，在此以 Oracle 資料庫系統為範例作說明：

表 4- 14：BASIC （拓片基本資料）

BID	ACCESS_NO	DB_TYPE	TITLE
1	00099	漢代石刻畫像	武氏北面畫像
2	188068-0006r	甲骨文	甲骨刻辭
3	186845	青銅器	洹子孟姜壺
4	01729	遼金元拓片	德州修碑樓堂事記
5	10747	佛教石刻造象	劉根四十一人浮圖

表 4- 15：BASIC 資料表說明

欄位名稱	欄位型態	欄位說明	索引名稱
BID	NUMBER	主鍵，流水號	PK_BID
ACCESS_NO	VARCHAR2	登錄號，NOT NULL	IX_ACCESS_NO
DB_TYPE	VARCHAR2	拓片類別	
TITLE	VARCHAR2	主要題名	IX_TITLE

表 4- 16：DIMENSION （拓片高廣）

DID	WIDTH	LENGTH	BID
1	20	30	2
2	40	50	1
3	25	40	3
4	30	34	5
5	20	67	4

表 4- 17：DIMENSION 資料表說明

欄位名稱	欄位型態	欄位說明	索引名稱
DID	NUMBER	主鍵，流水號	PK_BID

WIDTH	NUMBER	寬，cm	IX_ WIDTH
LENGTH	NUMBER	長，cm	IX_ WIDTH
BID	NUMBER	FK，外來鍵	

1. 使用正確型態搜尋

在搜尋條件中，若使用錯誤型態進行搜尋，雖有時資料庫會自動對錯誤型態進行轉換，但卻有可能無法使用索引。如下例，假設 ACCESS_NO 欄位型態為字串，則 SQL 撰寫時應使用單引號將字串包夾，若未使用單引號，資料庫會自動將「where ACCESS_NO = 00099」搜尋條件改為「where TO_NUMBER(ACCESS_NO) = 00099」，則無法使用索引來搜尋，只能使用表格掃瞄方法來搜尋。

較不好的寫法:

```
select * from BASIC where ACCESS_NO = 00099
```

較好的寫法:

```
select * from BASIC where ACCESS_NO = '00099'
```

2. LIKE 部份比對查詢

當使用部份比對時，會設定「%」符號，例如：字串'%畫像'表示某某畫像，「%」可用於字串首、字串中與字串尾，一般資料庫系統中，只有置於字尾的 SQL 才能使用索引，因為索引通常會使用多元樹(N-ary Tree)來建置（如 B+樹，請參考本章第一節第四小節），若字串前面不固定，將無法用此資料結構建立索引。

較不好的寫法:

```
select * from BASIC where TITLE like '%畫像'

select * from BASIC where TITLE like '武氏%畫像'

select * from BASIC where TITLE like '%北面%'
```

較好的寫法:

```
select * from BASIC where TITLE like '武氏北面%'
```

3. 欄位需運算之查詢

查詢條件中，若有欄位需經過運算，則無法使用索引，只能使用表格掃描方式。如下例，想找出寬度大於 300mm 的所有筆數時，通常會直接下「where WIDTH*10 > 300」，以上這種方式就無法使用索引，較好的方式是改成「where WIDTH > 300/10」。

較不好的寫法:

```
Select * from DIMENSION where WIDTH*10 > 300
```

較好的寫法:

```
select * from DIMENSION where WIDTH > 300/10
```

4. 複合索引（Concatenated Index）的使用

若要使用到複合索引，則查詢條件必須包含複合索引中的第一順位欄位。如下例，首先為拓片基本表格（BASIC）建立登錄號（ACCESS_NO）與主要題名（TITLE）之複合索引 IX_ACCESSNO_TITLE，第一順位欄位為 ACCESS_NO，若只使用「where TITLE = '洹子孟姜壺'」查詢條件時，因不包含 ACCESS_NO 此欄位，所以無法使用此複合索引。

建立複合索引 IX_ACCESSNO_TITLE

```
create index IX_ACCESSNO_TITLE on BASIC (ACCESS_NO,TITLE)
```

較不好的寫法:

```
select * from BASIC where TITLE = '洹子孟姜壺'
```

較好的寫法:

```
select * from BASIC
where TITLE = '洹子孟姜壺' and ACCESS_NO = '186845'
select * from BASIC
where ACCESS_NO = '186845' and TITLE = '洹子孟姜壺'
select * from BASIC
where ACCESS_NO = '186845'
```


5. Order By 的使用

查詢時，Order By 只能在「不能為空值」限制（Not Null）的欄位中，才能使用索引。如下例，主題（TITLE）欄位沒有非空值限制，登錄號（ACCESS_NO）有非空值限制，則排序登錄號（ACCESS_NO）時才會使用到索引。

較不好的寫法:

```
select * from BASIC order by TITLE
```

較好的寫法:

```
select * from BASIC order by ACCESS_NO
```

6. Group By 的使用

請注意任何採用 Group By 的 SQL 皆無法使用索引。

7. NULL 查詢

索引值中並不包含 NULL 值，所以當查詢條件含有 NULL 或 NOT NULL 時，是無法使用索引，但 NOT NULL 可用「＞ ‘’」代替，即可用到索引。

較不好的寫法:

```
select * from BASIC where TITLE is not null
```

較好的寫法:

```
select * from BASIC where TITLE > ‘’
```

8. 資料表結合

當需結合多個資料表做查詢時（如下面範例），查詢資料表的順序就相當重要，盡量將資料量小的表格置於前方、對資料量大的表格建置合適的索引欄位。

```
select B.ACCESS_NO,B.TITLE,D.WIDTH,D.LENGTH
from BASIC B,DIMENSION D
where BASIC.BID = DIMENSION.BID
```

9. 只查詢需要的欄位

在 select 欄位時，只選擇需要的欄位，避免使用「*」萬用字元。如下例，當想找出拓片基本資料（BASIC）中登錄號（ACCESS_NO）為 186845 的主要題名（TITLE）時，避免使用「Select *」，改用「Select TITLE」即可。

較不好的寫法:

```
select * from BASIC where ACCESS_NO = ‘186845’
```

較好的寫法:

```
select TITLE from BASIC where ACCESS_NO = ‘186845’
```

四、欄位索引技術

從資料庫中找到所需資料方式，基本上有「資料表格掃描」與「索引掃描」兩種方式。資料表掃描會逐一檢查資料表中所有記錄，找出符合搜尋條件的資料，另外一種方式則為本節所介紹之索引方式。索引為事先建立，搜尋時會先找索引，取得符合搜尋條件的資料表儲存位置，然後再根據此儲存位置，找出符合資料記錄，基本上有下列三種方式可建立索引：

（一）建立主鍵（Primary Key）

建立主鍵的目的為可區別出表格內的各個記錄（列），讓每一記錄皆有不重

複、不為空值的主鍵以供辨識。主鍵可為單一欄位，或複合式欄位，但一個資料表中，只能有一個主鍵。當主鍵建立時，通常資料庫系統會自動為此主鍵欄位建立索引。

（二）建立唯一限制（Unique）

當欄位有資料不重複需求時，可使用此限制。通常資料庫系統會自動為此欄位建立索引。

（三）自行建立索引

自行建立索引，設定所需索引欄位。

常見的索引結構包含「雜湊表」及「B+樹」等資料結構。雜湊表在概念上是透過設計一個雜湊函數，將索引值對應到一個雜湊值。每個雜湊值維護一個串列，指向所有對應到此雜湊值的表格資料。若要找某個索引值的資料，則利用雜湊函數算出其雜湊值，再到該串列上搜尋此索引值是否存在。雜湊表的效能通常取決於雜湊函數的設計，若過多不同的索引值對應到相同的雜湊值，將導致串列長度太長，以致於減緩搜尋速度。

B+樹是一種樹狀資料結構，能夠使資料保持排序的方式儲存，並允許快速的插入和刪除資料的動作，這些動作通常在對數(log)處理時間可以完成。下圖 4- 5 顯示 B+樹之樹狀結構的基本概念(Knuth, 1973)，圖中樹狀資料結構的節點負責儲存索引值及指標，所有節點可分為索引集合(Index Set)及序列集合(Sequence Set)：序列集合內的節點中，向下的指標負責指向實際的表格資料，向右的指標允許（依索引值的大小）由小而大的掃瞄所有或部份排序後的資料；而索引集合內的節點表示各種不同索引值區間。當要查詢擁有某個索引值的資料時，需從樹狀結構的根節點(Root)，由上而下搜尋，搜尋過程中，索引集合內的每個節點提供不同檢索值的範圍，例如：查詢數值 71 時，因 71 介於 50 和 82 中間，在第二層時需拜訪中間的節點，又 71 大於 70，故再拜訪剛才中間節點中，最右邊指標所指的節點，最後找到索引數值為 71 的資料。一般而言，資料庫中的資料非常龐大，為避免循序搜尋(Sequential Search)的問題，此樹狀結構提供非常有效的檢索方式，若圖中每一個節點的存取對應一次的資料存取，查詢 71 時，使用此樹狀結構只需 3 次存取動作，若考慮使用序列集合以進行循序搜尋時，就需要 6 次存取動作。

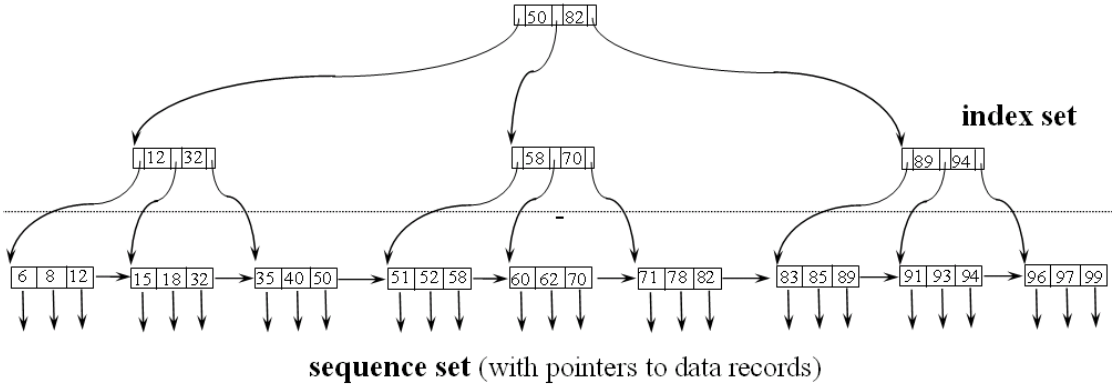


圖 4- 5：B+樹之樹狀結構

B+樹(Bayer, 1972)基本上是由 B 樹(Bayer, 1971)所演變而來，此處介紹的 B+樹為 Knuth 所提出的變形(Knuth, 1973)，此外還有 B*樹等其他變形。有興趣的讀者可參考資料庫相關書籍(Date, 2003; Silberschatz, Korth, & Sudarshan, 2005)。

五、數位典藏上其它檢索需求與問題

此節將探討數位典藏的「資料庫全文檢索需求」、「跨多個資料表格之檢索需求」及「資料庫之中文缺字問題」。

（一）資料庫全文檢索需求

有些數位典藏系統會將整篇文件儲存於一個欄位，此時除了資料庫需支援大型物件（Large Object, LOB）存取外，也需要支援文件全文檢索、符合關鍵字之加權排序、或關鍵字之邏輯運算（and、or、not）等功能。在目前常見的資料庫中，Microsoft SQL 與 Oracle 資料庫皆有內建全文檢索功能，使用起來會比較方便，若在無支援文件資訊檢索的資料庫中，就只能自行加掛外部全文檢索引擎(如 Lucene，請參考 4.2.8）以達到此功能。接下來，將以 Oracle 資料庫為例，簡單介紹如何使用 Oracle Text 全文檢索功能：

1. 挑選出需要進行全文檢索欄位，並為此欄位建立全文檢索索引

```
CREATE INDEX INDEX_NAME_IX
    ON INDEX_TABLE(INDEX_FIELD) INDEXTYPE IS CTXSYS.CONTEXT
    PARAMETERS ('LEXER WKSYS.WK_CHINESE_LEXER TRANSACTIONAL')
```

INDEX_NAME_IX：索引名稱

INDEX_TABLE：欲索引之「資料表名稱」

INDEX_FIELD：欲索引之「欄位名稱」

WK_CHINESE_LEXER：使用中文剖析方式

2. 使用“CONTAINS”函數查詢 INDEX_FIELD 欄位之 KW 關鍵字，並傳回 Oracle Text 所計算之分數大於 0 結果（分數範圍從 0 到 10），分數越高代表此欄位中關鍵字出現次數越多。

```
SELECT * FROM INDEX_TABLE WHERE CONTAINS(INDEX_FIELD,'KW')>0
```

INDEX_TABLE：已建索引之「資料表名稱」

INDEX_FIELD：已建索引之「欄位名稱」

KW：欲搜尋之關鍵字

3. 其他進階搜尋功能

使用“AND”布林運算子查詢關鍵字 (KW)

```
SELECT * FROM tableName WHERE CONTAINS(indexcontext,'KW1 AND KW2')>0
SELECT * FROM tableName WHERE CONTAINS(indexcontext,'KW1 & KW2')>0
```

使用“OR”布林運算子查詢關鍵字 (KW)

```
SELECT * FROM INDEXTABLE WHERE CONTAINS(indexcontext,'KW1 OR KW2')>0
SELECT * FROM INDEXTABLE WHERE CONTAINS(indexcontext,'KW1 | KW2')>0
```

使用“NOT”布林運算子查詢關鍵字 (KW)

```
SELECT * FROM INDEXTABLE WHERE CONTAINS(indexcontext,'KW1 NOT KW2')>0
SELECT * FROM INDEXTABLE WHERE CONTAINS(indexcontext,'KW1 ~ KW2')>0
```

使用“ACCUM”將 KW1 和 KW2 兩者分數相加查詢，列出分數>0

```
SELECT * FROM INDEXTABLE WHERE CONTAINS(indexcontext,'KW1 ACCUM KW2')>0
SELECT * FROM INDEXTABLE WHERE CONTAINS(indexcontext,'KW1 , KW2')>0
```

使用“MINUS”將 KW1 和 KW2 兩者分數相減查詢，列出分數>0

```
SELECT * FROM INDEXTABLE WHERE CONTAINS(indexcontext,'KW1 MINUS KW2')>0
SELECT * FROM INDEXTABLE WHERE CONTAINS(indexcontext,'KW1 - KW2')>0
```

使用“{}”將欲搜尋句子或保留字（EX:AND,OR,NOT……）括弧起來查詢

```
SELECT * FROM INDEXTABLE WHERE CONTAINS(indexcontext,'{PHRASE}')>0
```

使用“NEAR”搜尋相鄰關鍵字 KW1 及 KW2，相鄰越近分數越高

```
SELECT * FROM INDEXTABLE WHERE CONTAINS(indexcontext,'KW1 NEAR KW2')>0
SELECT * FROM INDEXTABLE WHERE CONTAINS(indexcontext,'KW1 ; KW2')>0
```

（二）跨多個資料表格之檢索需求

有些典藏系統需要提供使用者一個統一的查詢介面，同時搜尋非常多個資料表格，此時需同時查詢不同表格中的多個欄位，在某些複雜的典藏應用中，需查詢高達數百個欄位或數十個表格的資料，若採用傳統 SQL 中合併非常多個表格

的方法，將面臨效率不彰的問題。

常見的解決方法之一，是在資料庫中，建立一個專門查詢之表格，將所有不同典藏系統中要統一檢索的表格與欄位，集中於此表格中一欄位，以解決多個欄位與多個表格查詢時產生之複雜關係，並可增加查詢效率，此表格並紀錄每筆資料之相關資訊，以方便找出此筆資料來源之表格與欄位。此專門查詢之索引表格(Index Table)，初步分析所需欄位如下所示：

表 4- 18：索引表格所需欄位例示

ID	PrimaryKey	TableName	FieldName	IndexContext
1	12	tableA	fieldA	ABCDEFG
2	13	tableA	fieldB	EFGABCD
3	14	tableB	fieldA	GFEDCBA
…	…	…	…	…

ID：流水號，此表格之 PrimaryKey

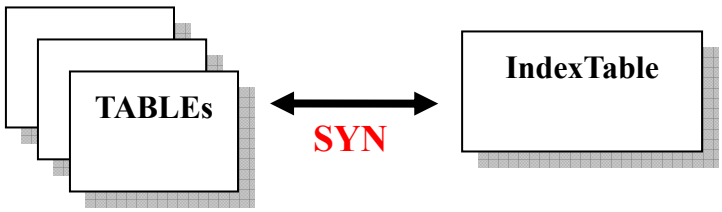
PrimaryKey　此筆記錄之「來源表格主鍵值」

TableName　此筆記錄「來源之表格名稱」

FieldName　此筆記錄「來源之欄位名稱」

IndexContext　欲檢索欄位之資料內容

此外，還需考慮的是原始資料表格與此索引表格同步化的問題，也就是當原始資料新增、修改、刪除時，此索引表格中資料也須適時作異動。較簡單的處理方式，是善用觸發(Trigger)程序，自動保持兩者間資料同步性，觸發程序可讓使用者自訂當監控之表格資料發生異動（新增、修改或刪除）時，所需觸發執行的動作。



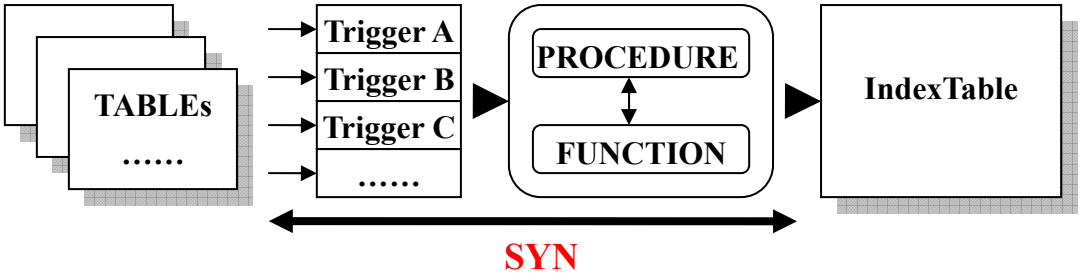


圖 4- 6：原始資料與索引表格資料同步化

以圖 4- 6 為例，最後每個需檢索的欄位，都會有一個觸發程序監控其新增、修改與刪除動作。若原始資料有異動時，就會自動將索引表格中的資料作同步更新。另外，為方便處理，可將各個觸發程序中新增、修改與刪除的功能抽離出來，另外撰寫成新的程序(Procedure)或函式(Function)，供各個觸發程序呼叫。

另一個問題是，要如何自動產生觸發程序。因為在此架構中，每個表格中需要被檢索的每個欄位，都會需要有一個觸發程序監控，當需要作全文檢索的欄位有上百個時，觸發程序同時也需要上百個，所以不太可能用人工去撰寫每一個觸發程序。在觀察每個監控的觸發程序後會發現，觸發程序間之差別只在於監控的表格與欄位不同，所以可以利用此特性，先將所有需要作檢索的相關表格、欄位資訊紀錄成一個表格，然後撰寫一個簡單的程式，讀取此表格後，自動化產生所有需要的觸發程序。同樣地，此表格也可額外設計操作介面，供使用者自行瀏覽、勾選資料庫中所需全文檢索的表格與欄位後自動產生，此表格建議欄位如下表 4- 19 所示：

表 4- 19：需檢索之表格與欄位建議列表

ID	TABLE_N	PKFIELD_N	FIELD_N	FIELD_T	INDEXTABLE_N
1	tableA	PKField_N_A	fieldA	Varchar2	IndexTableA
2	tableA	PKField_N_B	fieldB	number	IndexTableB
3	tableB	PKField_N_C	fieldA	Varchar2	IndexTableA
…	…	…	…	…	

ID　流水號 Primary Key

TABLE_N　此筆資料來源之表格名稱

PKFIELD_N　此筆資料來源之主鍵欄位名稱

FIELD_N　此筆資料來源之欄位名稱

FIRLD_T　此筆資料來源之欄位資料型態

INDEXTABLE_N　欲寫入之索引表格名稱(Index Table Name)，可用來規劃不同搜尋範圍）

當此架構完成後，接下來就可以使用 SQL 之檢索功能，針對索引表格中的 IndexContext 欄位建立各種索引（包含全文檢索)，建立完成後，即可利用此索引達到跨表格、跨欄位之檢索功能。除了 IndexContext 欄位，索引表格的其他欄位，則是用來幫助應用程式，找回此筆資料之原先所屬表格、欄位、主鍵等相關資訊，重建表格關係，以維持原先表格間之架構。目前此架構為數位典藏技術發展組（http://daal.iis.sinica.edu.tw/），實現跨表格及跨欄位之全文檢索的解決方案，但實際實作情況會依專案平臺與資料庫不同有所差異，而基本上解決方式的概念是相同的，於此特別提出以供讀者參考。

（三）資料庫之中文缺字問題

市面上常見資料庫，絕大部分都支援中文的搜尋，在數位典藏領域中，較特殊且需提出討論的是缺字問題（有關缺字問題的介紹，請參考第 3 章第六節），本節著重於資料庫之解決方案。

目前數位典藏技術發展組之缺字解決方案，是以「構字式」為基礎，並利用中央研究院資訊科學研究所文獻處理實驗室所發展的漢字構形資料庫來收錄數位典藏中所遭遇的缺字，並且在典藏系統中透過不同的處理流程，克服缺字資料在系統中著錄、顯示與查詢等問題，即使在網路環境下，仍然可以操作包含缺字之網頁，因此可做為典藏單位或相關技術人員缺字處理時之參考依循。

所謂構字式，就是從漢字字形結構的拆分與分析中，利用有限的部件及字根的組合方式，來表達任一漢字。例如「顥」，以構字式拆解的話，可拆分成「景」與「頁」兩個部件，其中為了表示部件與部件的連接關係，定義了三類，共計十三個的「構字符號」。「顥」的構字式為「景⿰頁」，因此構字式是由部件和構字符號組成，且「構字符號」也是一般文字和缺字的辨識依據。

當含有構字式的資料需被著錄時，由於構字式中的構字符號（如：⿰）無法被著錄，所以需被特別取出做特殊處理，如當著錄「景⿰頁」時，「⿰」符號會被轉換為跳脫字元(Escapes) 「」，所以在資料庫中「景⿰頁」會以「景頁」方式被存入，簡而言之，當此缺字需要被查詢或顯示時，也會以相同的原理相互作轉換，系統架構如圖4- 7：構字符號的轉換過程所示：

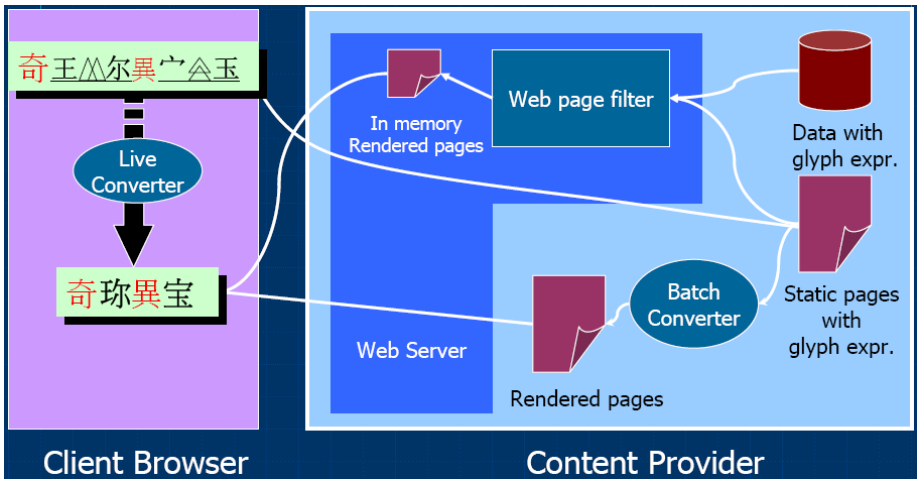


圖 4- 7：構字符號的轉換過程

六、常見資料庫系統

目前市面上常見之關聯式資料庫包含有 Oracle 資料庫、Microsoft SQL 資料庫、MySQL 及 PostgreSQL，茲分別簡介如下：

（一）Oracle 資料庫

商業資料庫，由專業資料庫廠商 Oracle 推出，一般常見功能皆具備，可說是目前市面上功能最齊全的資料庫。也因其功能眾多，其所提供之資料庫管理者介面相當複雜，甚至有些進階功能無法透過介面去管理，只能透過命令列（Command Line）方式進行設定，價格較為昂貴。其除可與 Java 做緊密結合，亦可於 Linux、FreeBSD、MS Windows、Solaris 等作業系統上執行，更多產品相關資訊可參考 oracle 之官方網站。

（二）Microsoft SQL 資料庫

商業資料庫，由 Microsoft 軟體公司推出，一般常見功能皆具備，價格上略比 Oracle 便宜，與 Oracle 不同之處在於，省略掉一些可讓資料庫管理人員自行微調的功能，以提供較友善之管理介面。其最大限制為，只能執行於 Microsoft Windows 作業系統上，更多產品資訊可參考其官方網站。

（三）MySQL

MySQL 基本上秉持 Open Source 精神，個人及非營利單位可免費使用，若用於商業用途，收費也相當便宜。所收取的費用，主要是用來協助 MySQL 研發所需。MySQL 雖沒有商用資料庫那麼多複雜的功能，但卻具穩定與容易使用的特性，可執行於 Linux、FreeBSD、MS Windows 等作業系統，適合經費有限，但卻有簡單資料庫需求的個人或非營利事業使用，更多產品資訊可參考 MySQL 之

官方網站。

（四）PostgreSQL

PostgreSQL 屬於 Open Source，由加州大學柏克萊分校(University of California, Berkeley)計算機系所開發，不管是個人或營利單位皆可免費使用，且很早就支援子查詢、View、Trigger、Procedure 等功能。在早期 MySQL 無法支援這些功能時，對有這些功能需求，卻又受限於經費的個人或營利事業來說，是最佳選擇，可安裝於 Linux、MS Windows、Solaris 等作業系統，更多相關資訊可參考 PostgreSQL 之官方網站。

七、資料庫檢索相關研究的資源

（一）重要的資料庫相關期刊

1. ACM Transactions on Database Systems (TODS)
2. ACM Transactions on Information Systems (TOIS)
3. IEEE Transactions on Knowledge and Data Engineering (TKDE)
4. VLDB Journal

（二）重要的資料庫相關會議

1. ACM International Conference on Management of Data (SIGMOD)
2. International Conference on Very Large Data Bases (VLDB)
3. Symposium on Principles of Database Systems (PODS) （較偏理論）
4. IEEE International Conference on Data Engineering (ICDE)

（三）重要的資料探勘相關期刊與會議

1. ACM Transactions on Knowledge Discovery from Data (TKDD) （新期刊）
2. ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)
3. IEEE International Conference on Data Mining (ICDM)

第二節　文件資訊檢索

文件資訊檢索旨在探討非結構化文件資料的檢索技術，搜尋引擎即為最常見的文件資訊檢索系統之一，其協助使用者快速搜尋非結構化的網路文件資料。相對於資料庫檢索的方法，文件資訊檢索方法中並無專家事先規劃好的固定欄位資料以供查詢，文件中出現的任意詞彙或主題都可能是使用者搜尋的目標。

一般而言，文件內容可藉由一組索引詞彙(Index Terms)或關鍵詞彙(Keywords)來表示，這些索引詞彙可直接以自動化的方式由文件內容抽取出來，或是由專家以人工的方式產生，而索引詞彙可視為該文件可能的檢索點，當使用者欲查詢符合某個主題的文件時，其通常輸入一個或多個與該主題相關的字詞，文件資訊檢索系統會比對這些索引詞彙與使用者查詢主題的相關性，進而計算每個文件與該查詢主題的相關程度，最後依照此相關程度，以由高至低地排列、輸出所有可能相關的文件。下圖 4- 8 以 Google 搜尋引擎為例，使用者輸入「數位典藏技術分項」查詢相關網頁，檢索系統依相關程度排序搜尋的結果。

圖 4- 8：Google 搜尋排序結果

由於索引詞彙是檢索比對的依據，為影響文件資料檢索成效的主要因素之一，愈多的索引詞彙表示文件有愈多的檢索點，愈能增加使用者可搜尋的範圍。然而對非常大量的文件資料而言，若考慮所有可能的索引詞彙，將會嚴重地消耗有限的電腦系統資源，因此，如何抽取有效且適量的索引詞彙，一直是此研究領域的重要課題。其中，過濾虛字、停用字(Stopword)或功能字(Function Word)為最常見的技巧之一，所謂虛字、停用字或功能字是指那些沒有承載語意的詞彙，例如：冠詞(on, the)、指示代名詞(this, that)、所有格(my, your)及連接詞(and, or)等，通常這些詞彙在文件集中，是屬於高頻詞彙；在英文文件中，另一種常見的技巧是詞彙正規化(Stemming)，由於英文字可能因不同時態而不同，此種方法用以解決有不同時態之英文字的問題。另外，如何將英文字的大小寫，與同義字的動詞、名詞甚至形容詞歸類為同義字等，亦是英文文件檢索中常見的相關問題。

相對於英文文件，中文文件最常碰到的問題是斷詞問題(Word Segmentation)。中文書寫中最小的單位是「字」(Character)，在中文文件中，只有字的界線而無詞的界線，無法像英文文件中每個詞(Word)之間會有空白間隔，來找出可能的檢索詞彙，因此，中文斷詞一直是中文文件處理與檢索中的重要研究議題之一。有關中文斷詞的問題與解決方法，請參考本章第二節第六小節。

為更清楚了解文件資訊檢索，以下將先比較「文件資訊檢索」與「資料庫檢索」的不同，再依序介紹常見的文件資訊檢索的效能評估方法、文件資訊檢索模型、文件查詢方法、文件索引技術、中文斷詞問題與搜尋引擎在數位典藏上的應用，並列出常見的文件資訊檢索系統及文件資訊檢索相關研究之資源。

一、文件資訊檢索與資料庫檢索的比較

相較於「資料庫檢索」,「文件資訊檢索」提供使用者完全不同的檢索模式；在資料庫檢索方法中，使用者需明確地描述其查詢語言，資料庫系統負責找出所有符合查詢條件的資料集合，這些輸出的資料集合都屬於「正確」的答案，並且沒有先後次序等不同重要性等級的區別；然而，在文件資訊檢索方法中，使用者以關鍵字詞或自然語言查詢的輸入方式，儘可能清楚且完整地描述其欲查詢的主題，或資訊需求(Information Need)，文件資訊檢索系統負責找出所有「可能」符合查詢主題的文件，並以排序的方式輸出結果，相關性越高的文件排在愈前面。

二、文件資訊檢索的效能評估方法

由於文件資訊檢索的結果，是依照與使用者查詢主題的相關程度排序輸出，除了執行速度的評比外，需要其他的方法來衡量其檢索效能，亦即衡量檢索結果滿足資訊需求的程度，其中最常見的衡量標準是回收率(Recall)和精確率(Precision)。回收率是計算所有相關文件被檢索出來的比例，精確率代表檢索到相關文件的比例，如下圖 4- 9 所示，其計算方式定義為：

回收率 ＝（ 檢索出相關文件筆數 / 所有相關文件的筆數 ） * 100%
＝(b / (b + d)) * 100%，
精確率 ＝（ 檢索出相關文件筆數 / 所有檢索出文件筆數 ） * 100%
＝(b / (a + b)) * 100%，

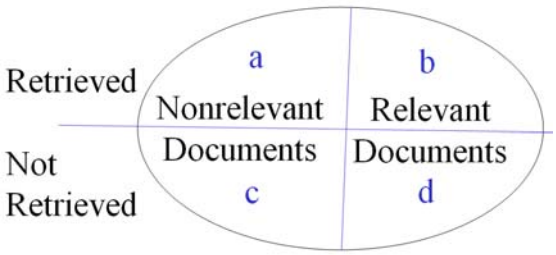


圖 4-9：文件資訊檢索的效能評估

精確率清楚反應出檢索結果的精確程度，但如果檢索出的筆數過少時，雖然可能有較高的精確率，仍不表示所有相關文件都會被檢索出來，此時，配合回收率可更有效地評估檢索效益。一般而言，回收率與精確率之間存在著反向關係，提高精確率通常會降低回收率，反之亦然。不同的應用環境，這兩者的權重可能不相同，以搜尋引擎為例，其檢索結果筆數通常太多，使用者又習慣只看排名前幾名的查詢結果，此時，精確率的重要性將高於回收率，因此，亦有結合回收率與精確率的衡量標準，如 F-Measure，藉由回收率與精確率不同的權重，F-Measure 可只以一個數值顯示檢索結果的優劣程度，關於其它衡量標準的細節，讀者可參考「Modern Information Retrieval」(Baeza-Yates, 1999)一書。

除了上述衡量標準外，在進行文件資訊檢索之效能評估時，另一個重要的問題是需要有公認標準的測試集，提供相關研究者一個規範化的環境與機制，來測量不同文件資訊檢索方法的效益，否則無法公平地判斷所檢索出的文件是否相關。Text REtrieval Conference（TREC）是第一個大規模的資訊檢索系統評估的學術研討會，1992 年首次舉辦，其建構的測試集包含龐大的測試文件、詳細的測試問題及多元的測試項目，之後，歐洲及亞洲各國的學者專家也籌辦不同語言的評估會議，並建構大規模的測試集，例如：歐洲各國聯合籌辦 CLEF(Cross-Language Evaluation Forum)資訊檢索評估會議，亦共同建構涵括多種語言的測試集；日本國立情報學研究所（National Institute of Informatics，簡稱 NII）舉辦 NTCIR (NACSIS Test Collections for IR) 評估會議，目前已建構主要亞洲語言包含中日韓文之測試集。

三、文件資訊檢索模型

為了檢索與使用者資訊需求相關的文件，資訊檢索系統需要某些方法來表示(Represent)文件、資訊需求及它們之間的關係，依照此表示方法，可以設計一排序演算法(Ranking Algorithm)產生所檢索結果，以決定任一文件與資訊需求是否相關；不同的表示方法產生不同的資訊檢索模型，而不同的資訊檢索模型有不同的預測模式，以決定哪些文件相關及哪些文件不相關。至目前為止，過去相關文

獻已提出許多種資訊檢索模型，本章節主要介紹兩種最基本的資訊檢索模型：布林模型(Boolean Model)及向量空間模型(Vector Space Model)。

此兩種模型皆視每一文件由一組索引詞彙所組成，索引詞彙可用以索引及摘要(Summarize)該文件的內容，就描述文件內容的功能而言，並非每一索引詞彙都具有相同的重要性，例如，如果一索引詞出現在每一個文件中，則其變成相對地不重要，因為當使用者以該索引詞查詢所有文件時，檢索系統無法有效分辨哪一個文件較相關，因此，不同的索引詞應該有不同的權重(Weight)以表示其在檢索上重要性，有了這個概念，茲依序介紹布林及向量空間模型如下：

（一）布林模型

「布林模型」是一個基於集合理論(Set theory)和布林代數(Boolean algebra)的模型，屬於一種精確匹配(Exact match)模型。在此模型中，僅考慮一文件內的某一索引詞是否出現，索引詞的權重值不是 1（出現）就是 0（未出現），使用者的查詢則以布林運算式(Boolean expression)表示，運算子包含 AND、OR 和 NOT。一文件如果滿足查詢條件的布林運算式，即被視為相關，例如，若有三文件分別包含某些索引詞，如下所示：

D1 = (information, retrieval, model)
D2 = (document, retrieval, application)
D3 = (modern, information, retrieval, theory, practice)

則 D1 及 D2 將滿足查詢的布林運算式 Q = (information OR document) AND retrieval AND NOT theory)。

布林模型的優點是規則簡單、清楚，並且容易實作。理論上，任一布林運算式皆可轉換成 DNF 型式(Disjunctive Normal Form)，亦即一種 E1 OR E2 OR E3 OR ... 的型式，其中 Ei (i=1,2,3,...)為 term1 AND term2 AND term3 AND... 的型式。由於權重值不是 1 就是 0，任一布林運算式可轉換成一組二進位字串(Binary string)，以加快電腦的運算速度。布林模型的最大的缺點是不支援部份匹配(Partial match)，在上述例子中，部份滿足查詢條件的文件 D3 被視為不相關，此外，布林模型還有其他缺點，例如，其產生的結果沒有排序，使用者可能不習慣以布林運算式的方式查詢，以及布林運算式的查詢結果往往不是筆數太多就是太少。

（二）向量空間模型

「向量空間模型」改進了布林模型的缺點，允許每個索引詞的權重值為一正數，而非只有 0 與 1 兩個值，這樣的改變讓文件與使用者查詢的相似度計算更具彈性。顧名思義，向量空間模型將文件與查詢以向量的方式表示，把文件資訊檢

索中，文件與查詢間相關程度的問題，轉換成向量空間中向量與向量間的相似度問題，最後，再以相似度的高低排序最後的檢索結果。

假設一文件集合 D={d_1, d_2, …, d_N}，共包含 M 個索引詞(t_1, t_2, …, t_M)，每個索引詞代表向量空間中的一個維度(Dimension)，文件 d_j 和查詢 q 的向量可表示成

$d_j = (w_{1,j}, w_{2,j}, \ldots, w_{M,j})$，

$q = (w_{1,q}, w_{2,q}, \ldots, w_{M,q})$，

其中 w_{ik} 表示第 i 個索引詞 t_i 在 k 中的權重值，k 可能是第 j 個文件或查詢 q，則向量 d_j 和向量 q 的相似度可利用兩向量間角度的餘弦(Cosine)值估算之，值愈大表示愈相似，如下圖 4- 10 所示：

$$sim(d_j, q) = \cos(\Theta) = \frac{\vec{d}_j \bullet \vec{q}}{|\vec{d}_j| \times |\vec{q}|} = \frac{\sum_{i=1}^{M} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{M} w_{i,j}^2} \times \sqrt{\sum_{i=1}^{M} w_{i,q}^2}}$$

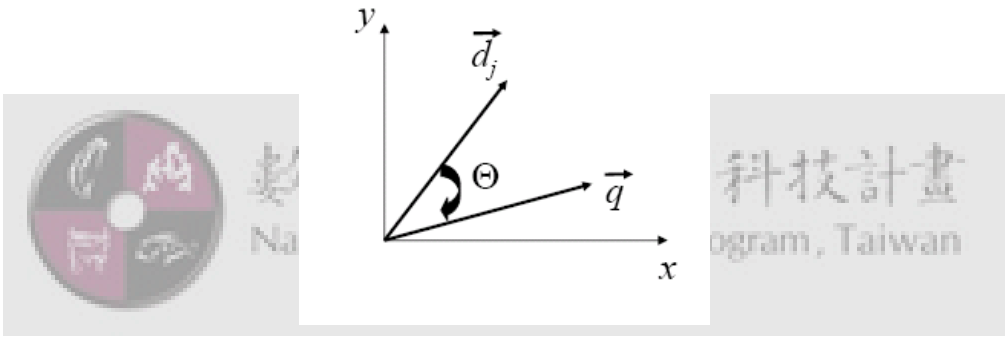


圖 4- 10：向量間的相似度計算

除了利用角度計算向量間的相似度，常見的相似度計算方式包括向量間的距離（如歐幾里得距離）或是投影後的距離。

向量空間模型面臨一個新的問題，亦即如何決定文件 d_j 和查詢 q 向量的權重值 w_{ik}。由於決定查詢 q 權重值的方法，通常類似於決定文件 d_j 權重值的方法，甚至更為簡單，在此，我們著重於如何決定文件 d_j 權重值的介紹。若考慮以角度估算的方式，則愈高的權重值對餘弦相似度計算影響愈大，換句話說，愈重要的索引詞需要愈高的權重值。一個衡量索引詞對某個文件的重要性的方式，是計算它在該文件出現的頻率，在此假設文件內頻率愈高的索引詞，表示其重要性也相對愈高，「索引詞頻率」（Term Frequency，簡稱 TF）即為常見的權重值估算方式之一。可是，高頻索引詞是否就一定重要？很顯然這個問題的答案是否定的，因為還得考慮其在所有文件中分布的情形，若一索引詞出現在絕大多數的文件中，則其不比另一僅出現在非常少數文件的索引詞來得重要，因後者更具有鑑別文件差異的能力，所以，另一個常見的權重值估算方式是「逆向文件頻率」(Inverse

Document Frequency，簡稱 IDF）。索引詞頻率 TF 指索引詞彙於單一文件中出現的次數，文件頻率 DF 則指出現該索引詞彙的文件個數；逆向文件頻率 IDF 與文件頻率 DF 為相反概念，索引詞彙的 DF 頻率過高，其 IDF 值就比較低，表示其出現在過多的文件中，導致缺乏有效的鑑別率；但由於 TF 指標對文件長度不一的情形誤差較大，文件長度正規化(Document Length Normalization)有助於改善此問題 (Singhal,Buckley, & Mitra, 1996)。過去有許多研究已證實，TF 乘以 IDF（TF×IDF）是用來判斷索引詞彙重要性的有效工具之一。

一個典型的 TF×IDF 權重值計算公式為(Salton, 1989)：

$$w_{ij} = tf_{ij} \times \log \frac{N}{n_i},$$

其中 w_{ij} 表示索引詞 t_i 在文件 d_j 中的權重值，tf_{ij} 是 t_i 在文件 d_j 中出現的次數，N 是所有文件個數，及 n_i 是指包含 t_i 之文件個數。

相較於布林模型，向量空間模型屬於一種最佳匹配(Best match)模型，其試圖在轉換後的向量空間中找最相似的向量，其優點包含權重值的設計方式改善了檢索結果，可依文件相關程度（或向量相似度）排序，可以支援部份匹配，並解決所有查詢條件都滿足才表示相關的問題；向量空間模型的缺點，為其假設所有索引詞間都互相無關(Independent)，而且權重值決定通常是依據經驗法則，較缺乏正規的理論基礎。

除了上述兩種模型，還有以機率理論為基礎之機率模型、乏晰集合理論(Fuzzy Set Theory)為主的乏晰集合模型，及語言模型(Language Model)等，相關介紹可參考探討資訊檢索之書籍，如「Modern Information Retrieval」(Baeza-Yates, 1999)、「Readings in Information Retrieval」(Jones & Willett, 1997)及「Information Retrieval: Data Structures and Algorithms」(Frakes & Baeza-Yates, 1992)。

四、文件查詢方法

文件查詢的方法大致上可分為四種：結構化查詢法、關鍵詞查詢法、範例式查詢法及自然語言之問答式查詢法，本節將依序介紹這四種方法：

（一）結構化查詢法

「結構化查詢法」旨在利用文件中特定的結構，以進行更有效查詢的方法。就像關聯式資料模型中的 SQL，利用表格的結構特性，可以建立複雜查詢。一般而言，文件亦有某些特定的結構，例如一篇新聞通常包含新聞標題及內文，一封電子郵件包含主旨、發信人、收件人、發信日期及郵件本文資料，一個網頁包含 URL、標題、修改日期、鏈結本文(Anchor text)等資料。結構化查詢即利用此

特定結構，查詢符合結構條件的文件。此類查詢需文件資料檢索系統事先剖析文件結構，並針對不同結構與內容提供檢索服務。

以 Google 為例，其進階搜尋功能允許使用者依網頁結構限制查詢條件，包含查詢網頁標題、內文、URL 或連結文字，例如，輸入「allintitle:李安 奧斯卡獎」，表示查詢標題內包含李安及奧斯卡獎的網頁。

（二）關鍵詞查詢法

「關鍵詞查詢法」是指使用者以一個或多個關鍵詞組合成一個查詢主題，是目前最普遍且最容易的查詢方法。此處關鍵詞可能包含「字」或「片語」(Phrase)，通常「片語」(如 information retrieval)比「字」(如 information AND retrieval) 可表達更精準的語意。提供關鍵詞查詢的檢索系統，往往會配合其他功能，例如：

1.布林查詢：以 AND、OR 與 NOT 的邏輯運算子限制查詢語意。

2.鄰近位置查詢：當輸入多個關鍵詞時，可限定各個關鍵詞在文件中出現的前後順序，例如 A*B*C 表查詢出現 A，再 B，最後 C 的文件，*在此表示可匹配零個至多個字詞。

3.同音（台大、臺大）、同義（中華文化、中國文化）或近似拼字的查詢：此類檢索系統需維護同音或同義詞典(Thesaurus)，或提供拼字檢查(Spell Checking)功能。

4.相關術語推薦：「臺大」的相關關鍵詞可能有「臺大醫院」、「臺大計算中心」、「臺大圖書館」等。當文件檢索系統蒐集大量的使用者查詢資料後，其可利用過去的查詢記錄，計算過去有哪些查詢詞彙與使用者輸入的查詢關鍵詞相關，以幫助使用者更快找到想要找的資料。

5.相關回饋(Relevance Feedback)：使用者依據前一次檢索所得的結果，指出傳回的資料中，哪些與查詢相關，哪些與查詢不相關，再將此訊息回饋給系統，逐漸導引系統搜尋的方向。

6.跨語言查詢：提供使用者查詢別種語言的文件，以英查中為例，使用者可輸入 Sony，查詢出現「新力」或「索尼」的文件。此類系統主要面臨兩個問題：一是某些字有多種翻譯，如英文字 bank 可能翻譯成銀行、河岸等，不同的翻譯往往有截然不同的意思。另一個問題是許多查詢詞，在翻譯字典中沒有出現，這最常發生在專有名詞與術語，如人名與組織名等，這些無法翻譯的字增加跨語言查詢的困難度。

（三）範例式查詢法

「範例式查詢法」是指使用者直接以某一文件查詢其它相似文件，欲找到相似文件，除了可以直接分析文件內容外，亦可利用文件間的結構關係，如網頁間的連結，或檔案系統階層結構等資源，評估文件間的相似度。範例式查詢法除了應用在尋找相似文件外，最普遍的應用是在尋找相似圖片，即以圖找圖，此功能對於沒有標記語意資料的圖片而言，提供更多的檢索機會。

（四）自然語言之問答式查詢

「問答式查詢」(Question Answering)允許使用者輸入問句，通常是自然語言式的問題(Natural language query)，提供使用者以較自然的方式來描述查詢主題，如「世界盃足球賽舉辦地點」。一般而言，問題類型可包含人物、地點、組織、事物、時間、數字等多類。為了快速處理此類查詢，在實際應用上，大部份的系統並不嘗試以複雜的自然語言理解方式來了解資訊需求，一般需先進行「問題分析」以取得重要的關鍵詞，然後取出文件中與這些關鍵詞高度相關的文句，再嘗試從這些文句中抽取及辨識出符合問句類型的候選答案，最後依候選答案的合適程度由高而低輸出。

五、文件索引技術

文件資訊檢索系統需事先索引文件以快速搜尋所需文件。在此介紹兩種常見的文件索引技術：「反向索引檔」(Inverted Index File)及「特徵檔」(Signature File)。基本上，文件索引方法與前述之資訊檢索模型或文件查詢方法無必然關係，一索引方法可能可以應用於多種模型。

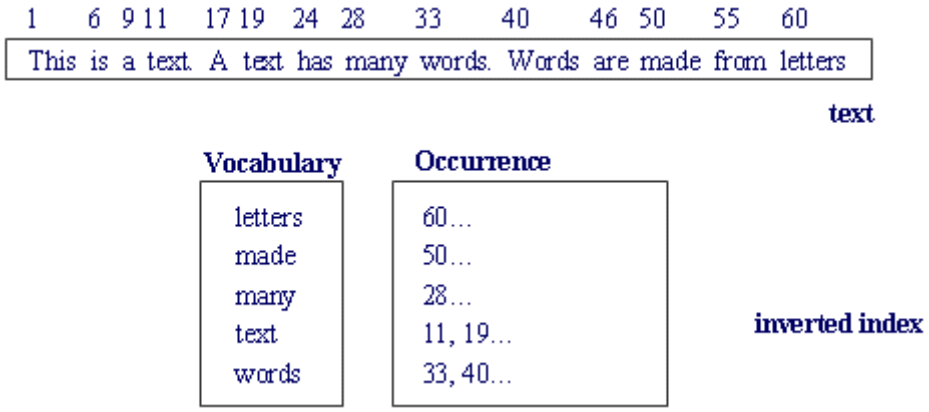


圖 4- 11：反向索引檔

（一）反向索引檔

「反向索引檔」旨在建立個別詞彙的索引，以上圖 4- 11 為例，圖中文件上的

編號表示某索引詞彙在該文件中的位置，例如 This 出現在第 1 個位置、Words 出現在第 40 個位置，反向索引檔是由兩元素所構成：詞彙表(Vocabulary)及出現(Occurrence)串列；詞彙表是每個文件中相異的索引詞彙集合（過濾虛字後），詞彙表中的每一索引詞彙指向一個出現串列，分別記錄該詞彙曾出現在哪些文件及文件中的哪些位置，一般而言，詞彙表所需的儲存空間比出現串列小很多，檢索系統會儘可能地把詞彙表存放在記憶體中，配合傳統的檢索技術，如雜湊表或以詞彙排序好的方式儲存，快速檢查某詞彙出現在哪些文件的哪些位置；相對地，出現串列可能需要存放在較大的儲存器，如硬碟，若要再省下出現串列所佔的空間，一種常見的方式是先將原文件切割成固定區塊，而出現串列以這些區塊為最小的索引單位，如下圖 4- 12 所示。

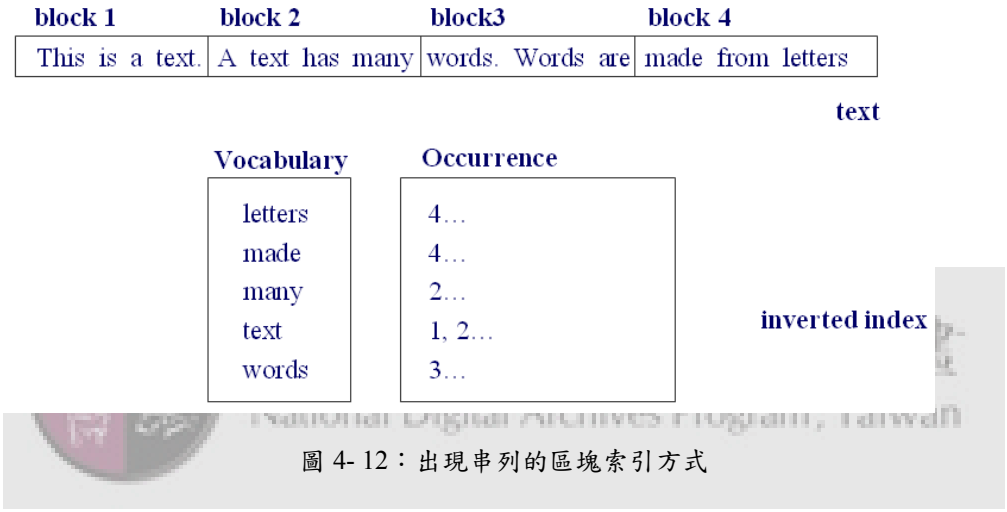


圖 4- 12：出現串列的區塊索引方式

出現串列記錄詞彙出現的位置，有助於「片語」及「鄰近位置查詢」，對於「關鍵詞查詢法」，反向索引檔可快速檢視有出現關鍵詞的出現串列集合，並可利用出現串列集合找出可能相關的文件。目前，反向索引檔仍是文件資料索引中最普遍應用的方法。

（二）特徵檔

「特徵檔」將個別索引詞彙編碼成 0 與 1 組成的二進位特徵向量，一文件的特徵向量則由該文件中所有索引詞彙的特徵向量所組成。由於特徵向量維度遠小於所有索引詞彙的個數，通常兩個不同的文件可能會對應到相同的特徵向量（類似雜湊表的基本概念）。應用特徵檔檢索時可分成兩個階段，第一階段經特徵檔運算，過濾掉不可能的文件，第二階段再把誤引(False Drop)的文件剔除。以下圖 4- 13 為例，一文件的特徵向量是由該文件中所有索引詞彙的特徵向量，以 OR 的運算計算出，若使用者查詢包含 Algorithm 與 Compression，則查詢的特徵向量為 100001 OR 010001=110001（任兩個對應位元，只要一個位元值為 1，OR 運算結果為 1），把查詢的特徵向量與所有文件的特徵向量做 AND 運算（任兩個對應位元，需兩位元值皆為 1，AND 運算結果才等於 1），得到文件 1 和文件 2 的

值與查詢的特徵向量相同，表示其可能包含有查詢詞彙，再分別檢查後，可發現文件 2 才是正確答案，此種方法非常適合檢索以「布林模型」建構的系統。

	Document 1		Document 2		Document 3
Keywords	Term Sig.	Keywords	Term Sig.	Keywords	Term Sig.
Algorithm	100 001	Algorithm	100 001	Database	001 001
Database	001 001	Image	100 010	Compression	010 001
Multimedia	010 010	Compression	010 001	Security	001 100
	111 011		110 011		011 101

圖 4- 13：特徵檔的編排

六、中文斷詞問題

中文書寫以「字」為單位，對映至語言單位的「詞素」(Morpheme)，比「詞」的單位還小。因此，中文文件中只有字的界線而無詞的界線。然而「詞」又是自然語言處理與資訊檢索的一個基本單位，如何將一篇文章正確的斷詞成為此類研究中一個重要的研究議題。

過去有許多研究提出不同的方法處理中文斷詞問題，其相關研究的分類與討論，在(Wu & Tseng, 1993)一文中有初步的比較與介紹。一般而言，過去提出的中文斷詞方法可分成三類：詞典法(Dictionary-based)、統計法(Statistical-based)及結合字典與統計的方法。

詞典法專注於找出文件包含哪些存在於詞典內的詞，此類方法需小心處理斷詞組合的岐義性(Ambiguity)問題，例如：文件中若出現字串「飛越南太平洋」的片段，有的系統會斷成「飛越」和「南太平洋」，而有的會斷成「飛」、「越南」和「太平洋」，目前最常使用的解決方法之一是最長匹配(Maximum Matching)法，假設給定一個 abcdefg 七個字的串列，若 a、ab、abc 都出現在詞典內，則選擇詞長最長者（即 abc)為一個詞，此方法有許多種改良版本，有的考慮文法、詞頻或字頻等其它資訊提昇幫助斷詞的正確率。詞典法為一簡單且有效的斷詞方法，然而此方法需要維護一個足夠大且動態的詞典，以避免無法辨識出那些詞典內未收錄的未知詞或新詞，這無疑需要大量的人力成本。未知詞主要包含人名、組織名、術語、專有名詞、簡稱、事件名稱等，在中文語言中，有些未知詞會遵循固定的命名規則，例如利用百家姓或職位頭銜可幫助人名的判斷，組織名可能稱為某公司、某院或某大學等，相關研究可參考中文的專有名詞辨識（Named Entity Recognition，NER)問題 (Chen, Ding, Tsai, & Bian, 1998)。因此，許多基於

詞典的斷詞法除了使用詞典斷詞外，亦加入偵測未知詞的能力(Chen & Bai, 1998)。

相反地，統計法不需依賴詞典，而是直接從一個很大的語料庫(Corpus)中學習中文詞彙的統計特性，以自動發現詞的邊界。常見的方法是考慮一個詞的內聚力與邊界變化情形，所謂詞的內聚力是指一個詞所組成的字之間應有較強的內聚力量，假設給定連續的兩個字串αβ，若在語料庫中顯示，當α出現時，β會常緊接著α出現，反之亦然，則αβ有較強的內聚力，例如「中正紀(α)」後面有非常高的機率會出現「念堂(β)」，因此，一詞其內部字串組合通常有較強的內聚力；從另一個角度看，一詞的前後文常需連接許多其它字詞形成一個句子，詞的邊界可連接的詞應有較大的變化，在上例中，若「中正紀(α)」後可接的字的選擇很少（很高的比例是「念」），則「中正紀」可能沒有包含一個詞完整的邊界，若「中正紀念堂(αβ)」後面可接的字很多，例如：中正紀念堂「的」總面積、中正紀念堂「位」於、中正紀念堂「座」落在等，則「中正紀念堂」可能有完整的邊界。由於統計法只需一個大的語料庫，不需人工編輯詞典，只要一個詞在統計上有明顯的重要性，很容易被辨識出，即使該詞未曾出現在詞典中，因此，統計法可以應用於未知詞的抽取，但統計法亦面臨其它問題，例如：對一個低頻詞，即該詞很少出現在語料庫，統計法往往無法有效正確辨識，另外，相較於詞典法，統計法所抽出的詞通常包含有較多的錯誤。以統計為主之中文斷詞法的相關研究可參考(Chien, 1997)。

（一）搜尋引擎在數位典藏上的應用

目前大部份的搜尋引擎都提供特定網站或網域的檢索功能，其有助於快速建構一具有檢索功能之數位典藏系統，以 Google 為例，使用者可輸入「**site:.ndap.org.tw 翠玉白菜**」，查詢在.ndap.org.tw 這個網域之所有電腦包含「翠玉白菜」的文件。因搜尋引擎會自動在網路上偵測是否有新的文件，並自動建立相關索引，對於一個公開的數位典藏系統，其建立檢索功能的成本最低，但相對地，此種方法也可能有安全上的問題，例如：是否有不想公開的典藏資料被搜尋引擎所自動索引，若要控制一網站何時及如何被搜尋及檢索，最安全的作法是不要公開網站內所有的檔案路徑，尤其是那些有安全或版權顧慮的檔案資料，另一種作法是在網路的最上層目錄（根目錄）設立一個特殊的檔案 robots.txt，在該檔案中，可以指定允許或封鎖哪些網頁可以被搜尋引擎檢索，例如在 robots.txt 設定：

User-agent: *

Disallow: /cgi-bin/

表示，所有的搜尋引擎都不可以存取該網站/cgi-bin/目錄及其子目錄的資料。robots.txt 設定的相關細節可參閱網頁 http://www.robotstxt.org/wc/norobots.html 。

七、常見文件資訊檢索系統

除了網路上常見的搜尋引擎外，以下列出一些開放原始碼的文件資訊檢索系統：

（一）Lucene 文件搜尋引擎

Lucene 是一個以 Java 程式語言開發的（文件）搜尋引擎函式庫(Library)，原創者 Doug Cutting 是一位資深的文件檢索專家，讀者可於其部落格找到許多相關介紹。Lucene 提供高效率的文件檢索功能，但不包含網頁的蒐集(Web crawling)及 HTML 格式的分析，若對 Lucene 內部技術有興趣，可參考「Lucene in Action」一書(Hatcher & Gospodnetic, 2004)。

（二）MG (Managing Gigabytes) 全文檢索系統

MG 全文檢索系統的目的是儲存、檢索和壓縮非常大量的文件（包含把實體文件掃瞄成影像檔案格式的文件），不但開放原始碼，作者也出版相關書籍「Managing Gigabytes」描述系統的設計與實作方法(Witten, Moffat, & Bell, 1999)，此系統目前已被紐西蘭數位圖書館計劃(NZDL, New Zealand Digital Library Project)所採用與維護，並依數位圖書館特別需求增加新的功能。

（三）ht://dig 網路搜尋引擎

ht://dig 提供一個完整網路搜尋引擎的功能，但僅適用於特定網站、網域或 Intranet 等較小規模的文件檢索目的，並不是要取代目前網站上常見的搜尋引擎如 Yahoo!或 Google，ht://dig 最初是聖地牙哥州立大學(San Diego State University)為了要搜尋校園網路內的網路伺服器而設計。

八、文件資訊檢索相關研究的資源

（一）重要的資訊檢索相關期刊

1. ACM Transactions on Information Systems (TOIS)
2. Journal of the American Society for Information Science and Technology (JASIST)
3. Information Processing and Management （前身為 Information Storage and Retrieval）

（二）重要的資訊檢索相關會議

1.ACM Annual International Conference on Research and Development in Information Retrieval (SIGIR)

2.ACM Conference on Information Knowledge Management (CIKM) （結合資料庫、資訊檢索及知識管理三個領域）

3.Text Retrieval Conference (TREC)

第三節　多媒體資訊檢索

由於多媒體資料亦包含結構化的資料屬性，如作者姓名、檔案建立時間等，及非結構化的資料屬性，如圖像內容描述，此類資訊可利用資料庫及文件資訊檢索方式搜尋。然而，圖像與影音具有獨特的媒體特性，提供使用者不同的視覺及聽覺效果，使用者可能想找相似圖像或曾聽過的新聞影音片段，因其有別於前兩種檢索技術，多媒體資訊檢索的相關議題，將於第 6 章「影音管理」探討。

第四節　總結

本章主要介紹「資料庫檢索」與「文件資訊檢索」兩種技術，並比較其相異之處。

「資料庫檢索」探討如何檢索結構化資料，以及簡介關聯式資料模型及結構化查詢語言，並深入剖析資料庫如何以最有效率的方式，執行使用者的查詢需求，進而說明在實務上設計結構化查詢語言時應注意的事項。「文件資訊檢索」探討如何檢索非結構化文件資料，簡介布林及向量空間模型，並說明除了簡單的關鍵詞查詢外，如何輔助其他功能以提昇文件查詢的效率與方便。

在索引方面，目前最常見的檢索技術在「資料庫檢索」與「文件資訊檢索」上並不相同，資料庫檢索的「樹狀索引結構」與文件資訊檢索的「反向索引檔」，是分別為兩種不同的檢索需求而設計的。

在應用檢索技術於數位典藏領域時，「資料庫檢索」強調如何支援全文檢索需求、跨多個表格的檢索需求及處理資料庫之中文缺字問題；「文件資訊檢索」則探討中文文件處理時的中文斷詞問題，及如何簡單利用現有的檢索系統於搜尋數位典藏內容上。

最後，本章亦簡介這兩種技術中常見的檢索系統，及相關研究的資源，以協助讀者進一步找到相關資訊。

參考網址列表

（National Institute of Informatics，簡稱 NII）舉辦 NTCIR (NACSIS Test Collections for IR) 評估會議，目前已建構主要亞洲語言包含中日韓文之測試集 http://research.nii.ac.jp/ntcir/

「Lucene in Action」一書(Hatcher, 2004) http://lucene.apache.org/java/docs/index.html

CLEF(Cross-Language Evaluation Forum) http://www.clef-campaign.org

Doug Cutting 的部落格 http://nutch.sourceforge.net/blog/cutting.html

Microsoft SQL 產品相關資訊 http://www.microsoft.com/sql

MySQL 產品相關資訊 http://www.mysql.com

Oracle 產品相關資訊 http://www.oracle.com/database/

PostgreSQL http://www.postgresql.org

Text REtrieval Conference（TREC）http://trec.nist.gov

紐西蘭數位圖書館計劃(NZDL) http://www.nzdl.org/ http://www.nzdl.org/html/mg.html http://www.cs.mu.oz.au/mg/

參考文獻

Baeza-Yates, R. & Riberiro-Neto, B. (1999). **Modern information retrieval.** Harlow, England: Addison-Wesley.

Bayer, R. (1971) Binary B-Trees for Virtual Memory. **SIGFIDET Workshop**, pp.219-235

Bayer, R. & McCreight, E. M. (1972). Organization and maintenance of large ordered indexes. **Acta Informatica**, 1(3), 173-189.

Chen, K.-J. & Bai, M.-H. (1998). Unknown word detection for Chinese by a corpus-based learning method. **International Journal of Computational linguistics and Chinese Language Processing**, 3(1), 27-44.

Chen, H.-H., Ding, Y.-W., Tsai, S.-C., & Bian, G.-W. (1998). Description of the NTU system used for MET2. In the **Proceedings of 7th Message Understanding Conference**. Retrieved Dec. 10, 2006, from http://www.itl.nist.gov/iaui/894.02/related_projects/muc/index.html

Chen, P. (1976). The entity-relationship model. **ACM Trans. on Database System,** l(1), 9-36.

Chien, L.-F. (1997). PAT-tree-based keyword extraction for Chinese information retrieval. **ACM SIGIR97**, pp. 50-58.

Codd, E.F. (1970). A relational model of data for large shared data banks. **Communications of the ACM**, 13(6), 377-387.

Date, C.J. (2003). **An introduction to database systems** (8th ed.). Addison Wesley.

Frakes, W.B. & Baeza-Yates, R. (Eds.). (1992). **Information retrieval: Data structures and algorithms**. Englewood Cliffs, N.J. : Prentice Hall.

Hatcher, E. & Gospodnetic, O. (2004). **Lucene in Action.** Manning Publications.

ISO/IEC JTC1/SC21 (1992). **Information technology- Database languages- SQL2**, ANSI, 1430 Broadway, New York, NY 10018.

Jones, K.S. & Willett, P. (Eds.). (1997). **Readings in Information Retrieval.** San Francisco, Calif.: Morgan Kaufman.

Knuth, D.E. (1973). **The art of computer programming**, Vol. 3 Sorting and Searchmg. Reading, Mass.: Addison-Wesley.

Salton, G. (1989). **Automatic text processing: The transformation, analysis and retrieval of information by computer**. Reading, Mass.: Addison-Wesley.

Silberschatz, A., Korth, H.F., & Sudarshan, S. (2005). **Database systems concepts** (5th ed). New York: McGraw-Hill.

Singhal, A., Buckley, C., & Mitra, M. (1996). Pivoted document length normalization. In the **Proceedings of the 19 th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval** , pp. 21-29.

Witten, I. H., Moffat, A., & Bell, T. C. (1999). **Managing gigabytes: Compressing and indexing documents and images**. Morgan Kaufmann Publishing.

Wu, Z. & Tseng, G.. (1993) Chinese text segmentation for text retrieval: Achievements and problems. **Journal of the American Society for Information Science**, 44(9), 532-542.

關鍵字列表

SQL 處理器（SQL Processor）
SQL 最佳化處理器（SQL Optimizer）
分離標準型式（DisJunctive Normal Form，簡稱 DNF）
反向索引檔（Inverse Index File）
文件長度正規化（Document Length Normalization）
片語（Phrase）
主鍵（Primary Key）
乏晰集合理論（Fuzzy Set Theory）
出現串列（Occurrence）

功能字（Function Word）
半結構化資料庫（Semi-structured Database）
卡式基（Cartesian Product）
布林代數（Bboolean Algebra）
布林運算式（Boolean Expression）
布林模型（Boolean Model）
交易（Transaction）
交集（Intersection）
全文檢索（Full-text Search）
列（Row, record, tuple）
向量空間模型（Vector Space Model）
合併（Join）
回收率（Recall）
多元樹（N-ary Tree）
字（Character）
自然合併（Natural Join）
自然語言式查詢（Natural Language Query）
行（Column, field, attribute）
序列集合（Sequence Set）
投影（Projection）
函式（Function）
歧異性（Aambiguity）
物件導向資料庫（Object-oriented Database）
表格（Table）
相關回饋（Relevance Feedback）
乘積（Product）
差集（Difference）
特徵檔（Signature File）
索引（Index）
索引掃描（Index Scan）
索引詞彙（Index Term）
索引詞頻率（Term Frequency，簡稱 TF）
索引集合（Index Set）
逆向文件頻率（Inverse Document Frequency，簡稱 IDF）
除法（Division）
停用字（Stop Word）
問答式查詢（Question Answering）
基於成本最佳化（Cost-based Optimizer，簡稱 CBO）

基於規則最佳化（Rule-based Optimizer，簡稱 RBO）
執行計畫（Execution Plan）
執行計畫流程（Access Path）
專有名詞辨識（Named Entity Recognition，簡稱 NER）
巢狀迴圈結合（Nested Loop Join）
排序合併結合（Sort Merge Join）
統計法（Statistical-based Method）
部分匹配（Partial Match）
最佳匹配（Best Match）
最長匹配（Maximum Matching）
循序搜尋（Sequential Search）
結構化查詢語言（Structured Query Language，簡稱 SQL）
結構化資料庫（Structured Database）
視界（View）
詞（Word）
詞典法（Dictionary-based Method）
詞素（Morpheme）
詞彙正規化（Stemming）
詞彙表（Vocabulary Table）
集合理論（Set Theory）
資料定義語言（Data Definition Language，簡稱 DDL）
資料表掃瞄（Table Scan）
資料控制語言（Data Control Language，簡稱 DCL）
資料操作語言（Ddata Manipulation Language，簡稱 DML）
資料檢索（Date Retrieval）
資訊檢索（Information Retrieval）
預存程序（Procedure）
精確匹配（Extra Match）
精確率（Precision）
語言模型（Language Model）
語料庫（Corpus）
複合索引（Concatenated Index）
選取（Selection）
聯集（Union）
斷詞（Word Segmentation）
雜湊表（Hash Table）
雜湊值（Hash Value）
雜湊結合（Hash Join）

關聯式資料庫（Relational Database）
關聯式資料模型（Relational Data Model）
關鍵辭彙（Keyword）
觸發程序（Trigger）
權重（Weight）
權限控管（Access Permission）