行政院國家科學委員會專題研究計畫　期中進度報告

資料串流上連續型查詢處理技術之研究(1/3)

計畫類別：個別型計畫
計畫編號：NSC94-2213-E-004-001-
執行期間：94 年 08 月 01 日至 95 年 07 月 31 日
執行單位：國立政治大學資訊科學系

計畫主持人：陳良弼

報告類型：精簡報告
報告附件：出席國際會議研究心得報告及發表論文
處理方式：本計畫可公開查詢

中　華　民　國 95 年 5 月 29 日

資料串流上連續型查詢處理技術之研究(1/3)

Research on Continuous Query Processing Techniques over Data streams

計畫編號：NSC－94－2213－E－004－001
執行期間：94 年 08 月 01 日至 95 年 07 月 31 日
計畫主持人：陳良弼
執行單位：國立政治大學資訊科學系

中文摘要

由於資料串流（data stream）迥異於傳統資料的特性，再加上眾多新興應用的推波助瀾，最近幾年來資料串流已逐漸成為新興應用中備受矚目的資料型態，舉凡資料串流管理系統（DSMS）的相關理論或應用雛型，皆為當前資料工程領域熱烈討論中的研究課題。本計畫將以研究 DSMS 的核心技術—連續型查詢（continuous query, CQ）之處理為主軸，發展此核心技術所需的三類關鍵技術：表格資料之連續型查詢處理（Relational CQ Processing）、查詢與資料串流之監控（Query and Data stream Monitoring），以及序列資料之連續型查詢處理（Sequential CQ Processing）。

對應於三類關鍵技術，在本年度計畫執行過程中，我們已完成第一年度預定完成之子技術，分別為可擴充式連續型查詢處理、查詢串流之樣型探勘、多數值串流之內容篩選。在利用查詢串流之樣型探勘支援可擴充式連續型查詢處理的研究中，我們將查詢分視為不同查詢樹，藉由頻繁子樹樣型探勘，找出各查詢間的共用子查詢，並分析單一子查詢的查詢規劃，進而推展出多查詢間可同時使用的全域執行規劃。另外在多數值串流之內容篩選方面，我們利用將查詢小片段分群，並利用滑動視窗將資料切成資料片段，透過資料片段和查詢小片段的群組間相似度計算，和序列資料本身的時序特性，提供刪除機制，加速近似結果的比對過程。

關鍵字：資料串流、連續型查詢、頻繁子樹樣型探勘、查詢規劃、序列資料

Abstract

Rapid advances in network commun-ications, software and hardware technologies bring huge amounts of data and form the data as continuous *data streams*. A data stream is an unbounded sequence of data persistently generated at a high speed. Due to its characters different from data stored in traditional databases and many applications relative to it, an enormous number of researchers pay attention to this research issue. At present, a new type of data management system, called data stream management system (DSMS), has become one of the most popular research areas in data engineering field. One of the kernel technologies in DSMS, namely *continuous query processing*, is developed in this project. The continuous query processing technology in this project is decomposed into three partitions including *relational continuous query processing*, *query and data stream monitoring* and *sequential continuous query processing*; each of them has been specified in the proposal of this project.

In the past one year, we have accomplished the purpose goals of the first year of this project. In the field of relational continuous query processing and query and data stream monitoring, multiple queries can be viewed as query trees, and relying on frequent sub-trees mining, common sub-queries can be found and applied to optimize global execution plan for these multiple queries. In the filed of sequential

continuous query processing, we propose a novel method based on *n-gram* to continuously process queries over event streams to find all approximate answers.

一、前言

傳統資料庫管理系統（DBMS）皆為建構於可永久存在的資料之上，對於這些資料，DBMS 可以提供穩定的儲存，並在查詢處理或是資料探勘的處理過程中，多次來回掃瞄資料庫，以達到精準而正確的結果。然而，在許多新興的應用中，資料被迅速而連續地產生並以串流的形式存在於應用系統中，導致傳統 DBMS 難以適用，此種特殊的資料型態一般被稱為資料串流（data stream）。由於資料串流迥異於傳統資料的特性，再加上眾多新興應用的推波助瀾，最近幾年來資料串流已逐漸成為新興應用中備受矚目的資料型態，舉凡資料串流管理系統（DSMS）的相關理論或應用雛型，皆是當前資料工程領域熱烈討論中的研究課題。本計畫將以 DSMS 的核心技術—連續型查詢之處理為主軸，發展此核心技術所需的三類關鍵技術：表格資料之連續型查詢處理、查詢與資料串流之監控，以及序列資料之連續型查詢處理，其內容分述如下：

一、表格資料之連續型查詢處理：對表格資料而言，在資料串流及傳統的資料型態上，都是以如何達成查詢最佳化為主要研究課題。由於在資料串流型態下，查詢及資料串流的分佈和流量皆會持續變動，若因流速過快則可能造成系統過載問題，因此本計畫將從可擴充性（scalability）、可調節性（adaptability）、和負荷卸載（load shedding）等問題分年著手考量。

二、查詢與資料串流之監控：監控查詢與資料串流而得的變動資訊，可提供查詢最佳化的分析之用。由於資料串流具有不定期更新和快速累積的特性，因此，我們需要有更有效率的資料分析技術，來幫助掌握查詢與資料串流最新的變化。本計畫分年探討查詢串流的樣型探勘（pattern mining）、資料串流的統計值概算、和資料串流的激變偵測（Burst Detection）等問題。

三、序列資料之連續型查詢處理：現階段已開發的 DSMS 乃承襲於傳統 DBMS 而來，只能用於查詢常見的表格資料，尚未支援序列資料的查詢；因此，本計畫將依照不同的序列資料型態分別進行研究、分年探討，一是針對多數值串流（Multi-valued Streams）發展的內容篩選技術，網路封包與電台音樂都屬於這類；二是針對多屬性串流（Multi-attributed Streams）發展的內容篩選技術，視訊影片即屬於這類；最後則同時考慮多個串流（Multiple Streams），這些串流可以是多數值或多屬性的資料型態。

二、執行進度規劃

本計畫擬開發之資料串流上連續型查詢處理技術其分年研究項目與各項目間之關連性如圖一所示。對於表格資料，第一年我們將利用各查詢之間的共通性推導最佳的執行規劃，發展可有效處理大量查詢的連續型查詢處理技術；第二年我們會加入資料串流動態特性考量，發展可隨著資料串流自行調整執行規劃的連續型查詢處理技術；第三年我們將進一步設計以服務品質為導向的負荷卸載技術，使得當系統資源無法滿足任何執行規劃時，DSMS 仍可透過犧牲最低限度的服務品質來維持運作。

在串流監控技術方面，第一年我們將研發頻繁子樹樣型（frequent subtree pattern）的探勘方法，用以提供前一研究主題獲取查詢間共通性資訊；第二年我們將開發資料串流上統計值的概算機制，用以監控查詢處理過程中流入與流出運算子的資料量，並提供前一研究主題計算各運算子需要分配的系統資源；第三年我們將探討激變偵測的相關課題，降低負荷卸載機制的負面影響，並積極地促進系統資源的重新分配，以提高系統整體利用率。

另外，對於序列資料，第一年我們將研

發多數值串流上的內容篩選技術，利用查詢小片段間的相似性和資料本身的時序性，加速近似結果的比對過程。第二年我們將觸角延伸到多屬性串流上的內容篩選技術，設計有效率的序列索引結構來儲存可能涵蓋不同屬性的查詢，並提出快速比對的演算法。第三年我們將整合前兩年的研究成果，進一步發展跨串流的內容篩選技術，用以滿足同時查詢多個異質串流的需求。

圖一：研究項目及其關係

對應於三類關鍵技術，在本年度計畫執行過程中，我們已完成第一年度預定完成之子技術，分別為可擴充式連續型查詢處理、查詢串流之樣型探勘、多數值串流之內容篩選。各項研究重點將會於下一節進行探討，以下為與本年度三項研究重點相關之國內外研究。

國內外相關研究

1、可擴充式連續型查詢處理

在資料串流環境下，單一查詢規劃之最佳化處理主要著眼於降低執行花費、記憶體需求及資料延遲時間。[VN02]在給定各資料串流的流速與各運算子的處理速度下，決定最佳的結合(Join)順序以滿足系統資源的限制。[BMMN04]則討論在多結合（Multi-Join）中如何決定各雜湊表格（hash table）的探查（probe）順序，以減低結合執行時所需之花費。[AN04]則提出滑動視窗中各運算子之花費估算模型，再根據估算所得，決定當下最佳的查詢規劃。

考慮大量連續型查詢處理的研究如NiagaraCQ[CDTW00] 與 TelegraphCQ[MF02][MSHR02]等計畫。NiagaraCQ系統為XML資料提供連續型查詢，主要方法是將相近查詢分組，同組查詢隸屬於同一個執行規劃，而執行規劃所得結果將是所有原來查詢結果的聯集，待結果產生後再根據群組分送給各查詢。其主要目標為提升XML查詢系統的擴充性，然而利用由下而上的方式尋找相似查詢則為其缺點所在，這樣方法將會遺漏許多可共同執行的子查詢。

而在Eddy[MSHR02]中，則將查詢規劃轉化為tuple的繞徑（routing）問題。各tuple依照不同順序流動，在輸出前會經由各運算子一次，繞徑的方式則依據最佳查詢規劃動態選擇。TelegraphCQ[CF02][MF02]利用Eddy的概念，將多個查詢處理簡化為查詢條件表格與資料表格之對稱雜湊結合（symmetric hash join）。然而，這樣的作法雖然提高了查詢規劃的可調節性，但是在各tuple繞徑時所產生的額外支出卻使系統整體效能下降。此外，該方法需針對每個資料來源維持獨立的查詢條件表格與資料表格，如此一來所需的記憶體需求將遠高於傳統查詢規劃。

2、查詢串流之樣型探勘

串流資料探勘的研究近年來備受注目，其中以資料項目的計數（counting）及其延伸出的頻繁樣型探勘等問題更為受重視。[MM02]利用O(logN)的空間複雜度，提供長度為N的串流資料中頻繁樣型的計數值，並確保其值在指定的誤差範圍內。[YCL04]應用Chernoff Bound理論來面對上述問題，將記憶體需求縮減到常數複雜度；此外，其答案集合都是由頻繁樣型所構成，但可能會遺漏部分答案。[AM04]則探討於滑動視窗中探勘頻繁樣型的問題。利用較複雜的結構，以空間換取滑動視窗內資料計數的準確性，此架構亦延伸至串流資料Quantile統計值的探勘問題，並獲得良好的效果。

目前串流資料探勘研究，針對較複雜的結構如樹狀結構的研究仍非常有限。[AAKA02]提出一個線上演算法StreamT來分析資料串流上的有序樹（ordered tree），而

StreamT維護記憶體的方式，無法保證其所得結果的正確性，亦無法提供可能的誤差範圍保證。[YLH04]則研究資料串流上有根有序樹（rooted ordered tree）的探勘，此意味串流上的資料必須有相同的根節點，進而求得具相同根節點之頻繁有序樹；然而，針對有根樹的探勘模式只能用在有限的特定應用上。

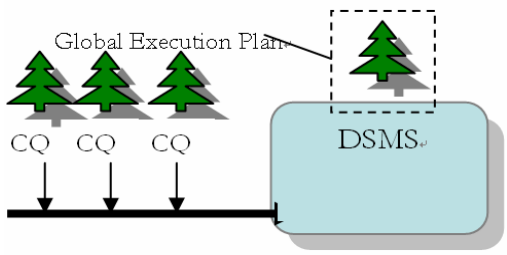
3、多數值串流之內容篩選

針對多數值字串比對的問題，目前的應用集中於對複音音樂之查詢，因大部分的音樂查詢系統乃是以音高為比對的標準，而在複音音樂中同一個時間點會有一個以上的音符同時發生，因此一首複音音樂可以由一個多數值字串來表示。對於複音音樂之查詢，[LP00]根據shift-or演算法發展一新演算法，但是與shift-or演算法相同，其查詢樣型的長度受限於計算機中的字元長度，而且其查詢只限於單音的形式；為了提供複音查詢的能力，[CE00]中提出以集合的方式儲存樂曲中音符出現的時間與該音符的音高，形成類似總譜表（score-like）的格式，依此來處理音樂查詢，但是這種技術只有在提供精準答案時具有效率；此外，[Dov99]則是提出一個以Dynamic Programming為基礎之演算法，希望能夠在音樂查詢時提供近似答案，然而由於該技術的限制，在音樂資料庫過大或是音樂字串過長時，會因為計算量過大而導致效率明顯降低。

三、研究方法、進行步驟及執行進度報告

在資料串流的環境中，除了資料本身以串流的形式進入系統外，各使用者所下的查詢，也可視為一種串流形式，如圖二所示；當多個使用者所下的查詢同時存在於系統之中，如何透過利用查詢間的共同子查詢來降低系統資源的需求，並增加結果的產出是本計畫的重點研究項目之一。另一方面，一個查詢可能會有多個查詢規劃（Query Plan），不同的查詢規劃可能會影響我們所找出來的共同子查詢；因此，如何找出適合所有查詢的全域執行規劃（Global Execution Plan）也是另一個研究重點。

當使用者下一個查詢之後，我們可將該查詢視為一棵查詢樹，樹根（root）與葉子（leaf)分別代表著查詢結果輸出與輸入的資料串流。其餘在樹中的節點（internal node）代表運算元，而樹中的邊（edge）則代表兩個運算間輸出與輸入的資料串流。如此一來，我們可將偵測共同子查詢的問題，轉化成在樹狀的資料串流（tree data stream）上探勘頻繁子樹樣型（frequent subtree patterns mining）的問題，之後再根據資料探勘所得的頻繁子樹樣型來產生全域執行規劃。本年度以查詢串流之樣型探勘支援可擴充式連續型查詢處理為方向，分別研究兩項重要技術，「全域執行規劃」及「頻繁子樹樣型探勘」，以及在多數值串流之內容篩選研究中，發展出「事件串流環境下之近似比對」，此三項執行成果分述如下。



圖二：查詢串流

<u>全域執行規劃之成果報告</u>

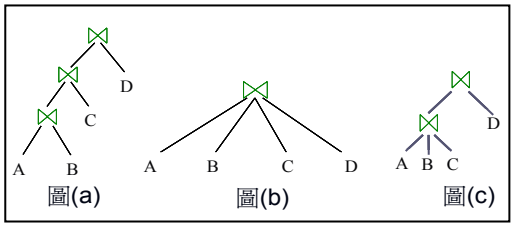
1、單一查詢查詢規劃最佳化之研究

研究目的

在資料串流環境下，已知的結合技術中，多個資料串流的結合有下列幾種查詢規劃，分別是一、二元結合 (Binary Join)如圖三(a)，二、多元結合如(M-Join)圖三(b)，和三、綜合結合如圖三(c)。在不一樣的網路環境中，如流速(Input Rate)以及結合率(Join Selectivity)的不同，會對不一樣的結合方式產生差異的處理時間。為了因應資料串流的龐大資料量，我們必須根據不一樣的流速及結合率找到處理時間最少的查詢計畫，此動作稱為查詢最佳化(Query Optimization)。

在做最佳化之前，我們必須先定義何為

最佳。在已知流速及結合率下，我們定義查詢規劃的費用(Cost)為處理每單位時間流進的所有 tuples 直到產生結合結果的時間。如此一來，最佳化的定義就是在所有查詢規劃中找到費用最小的規劃，即最佳規劃(Optimal Plan)。



圖三：多個資料串流的結合方法

研究方法

步驟一、窮舉法(Exhaustive Method)：窮舉法乃是根據資料串流的數目，產生所有可能的查詢規劃。我們先取兩個串流加以結合(此時是二元連接)，再取另一資料串流(1)加入到結合子中(Join Operator)，也就是形成多元結合；(2)再與此查詢中的任一串流連接，形成二元結合。對所有資料串流分別以這兩種方式產生新的查詢計畫，直到所有資料串流都已經加入即可產生所有的查詢規劃。對於所有產生的查詢規劃我們皆計算其費用，而後找出最佳規劃。根據實驗，雖然窮舉法可以找到最佳的查詢規劃，但是最佳化所需的時間以及記憶體最多只允許處理 6 至 7 個資料串流。

步驟二、貪婪法(Greedy Method)：在窮舉法中，我們列出了所有的計畫，加以運算其費用，導致處理時間過長及記憶體使用量龐大。針對這個情形，我們提出貪婪法加以避免。在產生查詢規劃的步驟上，對在加入資料串流後所產生的所有可能查詢規劃，根據各個尚未加入之資料串流的流速及結合率先行計算出所有計畫的期望費用(Expected Cost)，並只針對最小期望費用的可能規劃，產生新的查詢規劃。當我們運算完所有資料串流得到一個查詢規劃的實際費用，即可將此規劃的費用訂為臨界值(Threshold)，之後在產生其他規劃的過程中，可以此臨界值來決定某可能的查詢規劃是否應該繼續被考慮。以此方法到最後可得到近似最佳計畫(Near Optimal Plan)。根據實驗，貪婪法找到的查詢規劃都相當近似最佳規劃，而且可以用於 20 個資料串流以上的環境。因此，我們提出的方法可以在時間及空間上有效的解決多資料串流連接的問題。

2、感測器網路下之多查詢處理技術

研究目的

除了原本僅考慮資料串流環境外，由於感測器網路（sensor network）乃是一資料串流環境中之重大應用；因此，為了與真正實際應用結合，我們將多查詢處理問題延伸至建構在感測器網路上之資料串流研究。在感測器網路環境中，使用者可在任意節點發出其查詢。而多個查詢間，常有相互疊蓋（overlap）的情形；因此如何將多個相似查詢整合，將是查詢系統的關鍵性技術。有鑒於此，我們設計一多查詢最佳化技術(Multi-Query Optimization)，俾增進整體感測器網路之效率。我們所發展的技術其主要貢獻有二，分述如下：

第一，網路間（In-Network）查詢處理：將環境延伸至建構在感測器網路上的資料串流研究需額外考慮感測器（sensor）的原有特性，有別於現存採中央處理架構的資料串流管理系統(DSMS)，網路間查詢處理能減少網路中不必要的傳輸與計算，以節省感測器資源，如電力之使用。

第二，可擴充式連續型查詢處理(Scaleable Continuous Query Processing)：系統資源的限制是處理串流資料最大的濫觴。因此，我們的技術有系統地推導最佳化的全域執行規劃，以在有限的系統資源限制下，取得最大的系統資源利用率。

研究方法

步驟一、單一查詢規劃之產生（Generation of Individual Query Plans）：對於每一個使用者所下達的查詢，我們首先根據查詢內容與當下資料串流的統計資訊，分

析所有可能的查詢規劃，並設計一個單一查詢最佳化演算法，選擇出最佳的單一查詢規劃。根據傳統查詢最佳化方法所歸納的規則，我們將會在本步驟中重新考慮，但改以串流特性及運算結果的分享為主要考量。

步驟二、全域執行規劃搜尋空間之建立（Search Space Construction)：接著，我們對上個步驟所產生的多個查詢規劃，尋找合併執行的方法。其中最簡單的方法為：只考量以個別查詢的最佳化規劃，來進行分享執行(sharing execution)。然而，這樣的方法，並不一定能找到最佳的全域執行規劃。其主要原因為：這樣的方法，限制了個別查詢規劃間，可以進行分享執行的可能性。從實際的觀察中可得知，若我們對各查詢使用其子最佳化查詢規劃(sub-optimal plan)來進行分享，往往可以得到全域最佳的執行規劃。有鑑於此，我們針對所有可能的查詢規劃分享，進行考慮，並有系統地建立起一個全域最佳查詢規劃的搜尋空間(Search Space)。進一步地，我們將搜尋全域最佳的執行規劃問題轉化成為一個零與一的整數規劃問題(Zero and one Integer Programming)。

步驟三、全域最佳化執行規劃之搜尋空間（Searching for global optimality）：由於零與一的整數規劃問題，可證明為一個NP-hard的問題。因此，針對我們所建立的搜尋空間，我們主要提供兩個演算法來求得全域最佳的查詢執行規劃:

1. 較小限度演算法（Lower Bounding Algorithm）：較小限度演算法可確切的求得全域最佳化的查詢執行規劃，但可能需要指數時間的運算量。

2. 貪婪演算法（Greedy Algorithm）：基於較小限度演算法可能需指數運算時間，為求加速運算，我們再提出以貪心原則為基礎的貪婪演算法。貪婪演算法雖不一定每次皆可求得全域最佳化的查詢執行規劃。然而，根據我們的實驗可得知，貪婪演算法多可求得近似於全域最佳的查詢執行規劃，而且貪婪演算法保證僅需要多項式時間的運算量。

<u>頻繁子樹樣型探勘之成果報告</u>

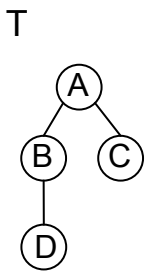
1、 頻繁子樹樣型探勘

研究目的

在資料串流環境中探勘頻繁子樹樣型，必須先釐清兩個問題：第一，某些頻繁子樹樣型可能因為有較長一段時間沒有出現在目前的資料串流中，其計數(count)雖然不變，但由於資料串流長度變長的影響，而使其出現比例（Support）相對下降、不足門檻值（threshold)，造成原本為頻繁子樹樣型轉變為非頻繁子樹樣型。另一方面，非頻繁子樹樣型也可隨著其計數的累加，當其出現頻率大於門檻值時，轉變為頻繁子樹樣型。若將所有子樹樣型的計數皆儲存下來，會使得系統記憶體負荷過大，但如果只儲存頻繁子樹樣型，並無法解決上述兩個問題；因此，如何列舉所有子樹並設計一個高效率的資料結構來儲存頻繁子樹樣型,即是本研究的重點。

研究方法

步驟一、樹之前置處理與編碼：我們所探討的查詢串流是由多筆查詢樹組成，查詢樹是一種有項目標幟無序樹（labeled unordered tree)，有項目標幟無序樹上的節點（node）具有下述特性：每個節點具有唯一的項目標幟（unique label），同一父節點（parent node）的所有節點沒有次序之分。為了避免同一棵無序樹產生大量的同構子樹（isomorphic subtree)，增加儲存和計數上的複雜度以及系統負載度，我們先將每棵查詢樹置入一正規化程序，其過程如下：即從樹根開始，依廣度優先順序與字元次序對該節點的所有子節點排序，所得之有序樹即稱為正規樹。同時為了便於產生所有子樹及計數等步驟，正規樹需再經過轉換成為特殊編碼的字串形式。此編碼方式僅能得到唯一的編碼字串，同時亦可將編碼字串轉換回原本的正規樹，此編碼過程為：依深度優先順序走訪正規樹中的各節點，依此順序列出其項目標幟以形成字串，如圖四所示。



編碼方式：

A1B2D3C2

圖四：有項目標幟的有序樹 T 及其對應編碼

步驟二、列舉所有子樹並合併至全域字首樹（Global Prefix Tree）：在此步驟，我們可根據編碼字串的特性，制訂的增加字首樹（APT, Augmented Prefix Tree）。其實因為在編碼的設計過程中，我們有將各節點所在的高度（Level）明確的紀錄下來，再加上編碼的過程是利用深度優先（depth first）順序，因此以這兩項特性反推我們可以建立一增加字首樹用以快速列舉並儲存所有的子樹樣型，之後再將其合併（merge）至全域字首樹中。

步驟三、以 Lossy Counting[MM02]演算法為基底，刪除計數非常小的子樹樣型：若我們將每棵查詢樹的增加字首樹找出，合併至全域字首樹，在資料流經一段時間後，可能會有非常龐大的全域字首樹，造成記憶體空間不足；因此，我們將全域字首樹的建置值基於 Lossy Counting[MM02]演算法上，當固定幾棵查詢樹加入全域字首樹之後，就將計數不足其規定的節點加以刪除，以達到節省記憶體之原則，並維持少算的部分能在容忍誤差範圍內。

步驟四、頻繁子樹探勘處理：當使用者要求回傳目前頻繁子樹時，我們對全域字首樹作深度優先的走訪，依據該子樹的計數值佔整體資料串流大小之比例是否超過使用者給定之最小支持度（Minimal Support），回傳頻繁且被標記為封閉的節點。

2、封閉子樹樣型探勘

研究目的

延續頻繁子樹樣型探勘之研究，我們將頻繁子樹樣型探勘的問題，延伸至封閉頻繁子樹樣型探勘（closed frequent subtree pattern mining）。封閉子樹樣型的優點在於其個數小於所有的子樹個數，且依然保存所有子樹的完整資訊，所以我們只要探勘出封閉頻繁子樹樣型，即可利用其封閉特性，推得所有頻繁子樹樣型；同時，由於探勘封閉頻繁子樹樣型對記憶體的需求小於探勘頻繁子樹樣型，因此在記憶體空間限制下，探勘封閉頻繁子數樣型來推得頻繁子樹樣型會是較符合空間限制之解法。

研究方法

步驟一、樹型編碼及列舉所有子數：仿造頻繁子樹樣型探勘作法，當一個新的查詢樹進入系統當中後，首先我們先對這棵樹進行正規劃及字串轉換得到相對應的編碼字串, 再根據此編碼字串利用增加字首樹列舉出相對應於此樹之所有子樹樣型的編碼字串。

步驟二、標記封閉子樹：根據增加字首樹的設計，右邊的路徑是其左邊路徑的子集，最左路徑(left most path)中的每一個節點只被其子節點所包含，而最左路徑葉子(left most path leaf)不被其他任何節點所包含，因此當我們根據深度優先的順序將增加字首樹合併入封閉全域字首樹（Closed Global Prefix Tree）時，我們可以不做任何檢查就直接將最左路徑所對應到封閉共同字首樹節點的計數做累加，同時將最左路徑葉子所對應的節點標記為封閉子樹。

而增加字首樹中其它的路徑的節點是否要併入封閉全域字首樹中，乃取決於該節點是否被其它節點包含。如果在封閉全域字首樹中找不到與其相對應的節點，則表示該節點本身以及其子孫節點所代表的子樹都不為封閉子樹。但是由於資料一直流入，該子樹可能隨著新資料的流入在未來成為封閉子樹；因此我們利用建立封閉連結（closure link）來解決此一問題，快速復該子樹的計數（count）。

步驟三、利用刪除機制避免記錄計數太小之封閉子樹：仿造上述頻繁子樹樣型探勘

的作法，為因應資料串流的環境特性，所以我們將封閉全域字首樹套用 lossy counting [MM02]演算法提供必要的少算動作，使其能以更少的空間記錄封閉子樹資訊，但仍能於一定的容忍誤差範圍內概算出所有封閉頻繁子樹的計數值。

步驟四、封閉頻繁子樹探勘處理：當使用者要求回傳目前封閉頻繁子樹時，我們對封閉全域字首樹作深度優先的走訪，依據該子樹的計數值佔整體資料串流大小之比例是否超過使用者給定之最小支持度（Minimal Support），回傳頻繁且被標記為封閉的節點。

事件串流環境下之近似比對之成果報告

研究目的

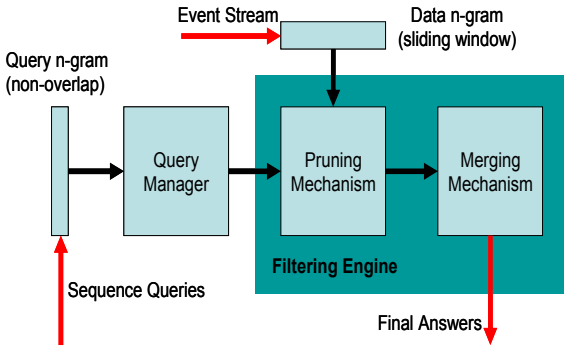
由於與資料串流相關之應用快速增加，因此有許多資料串流處理系統與相關的技術便孕育而生，希望能在資料串流的環境中，得以有效率地處理大筆流入的資料，同時也能夠支援連續性查詢的需求。

為了能夠處理更複雜的資料串流與查詢類型，以支援其他重要的應用，有別於以往只考慮表格資料之查詢處理，在本研究中，我們除了考慮表格資料外還增加了比表格資料更複雜的序列資料之考量。事件串流（event stream）即可視為一種序列資料的資料串流。在事件串流中，每一個事件可以包含不同個數的值；因此事件串流可以用來處理更為複雜的多媒體資料或是網路封包等；同時，在事件串流下，使用者通常感興趣的是某一段連續的事件是否發生，因此我們定義了一種新的查詢類型稱之為序列查詢（sequence query），讓使用者能夠持續偵測某一段連續事件是否出現。

有時候使用者並不能下達完全精確的查詢，或者是資料串流在傳輸的過程中會有資料誤差的情況發生，因此我們必須要提供每一個查詢能夠獲得近似答案而非只有精確答案，如此一來才能夠符合實際應用的需求。

研究方法

步驟一、將資料與查詢分別前處理為資料 n-gram 與查詢 n-gram：為了降低系統的運算量以符合查詢的及時需求與系統資源的限制，我們所發展的事件序列篩選系統採用之根本概念乃是利用查詢間存在的共通性，讓相似的查詢能夠一起進行比對的工作，進而節省運算工作量。圖五為我們系統運作的流程。查詢管理者（Query Manager）的工作是對所有註冊的查詢建立索引架構，我們將每個查詢分割成數個不重疊且長度為 n 的片段，稱為查詢 n-gram（query n-gram）再套用一分群方法將相似的查詢 n-gram 放在同一分群之內。同時，為了加速查詢比對的工作，每一個分群都會建立一個摘要（summarization）以代表該群，作為和每一筆流入資料比對之依據。另外，我們利用一流動視窗（sliding window）擷取長度為 n 的資料片段，稱為資料 n-gram（data n-gram）。當查詢索引建立完成後，刪除機制（Pruning Mechanism）會開始針對事件串流所流入的每一筆資料 n-gram 進行比對，以找出哪一筆資料 n-gram 有可能成為某一查詢答案的一部份。



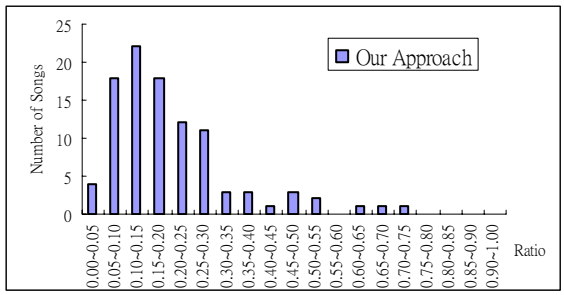
圖五：事件序列篩選系統之流程圖

步驟二、資料 n-gram 和查詢 n-gram 的比對：我們提出了一個計算資料 n-gram 與每一分群摘要間限制編輯距離(restricted edit distance）的方法，讓系統得以快速地驗證目前流入的資料 n-gram 一定不會是哪些分群的答案。如果一個資料 n-gram 與一個分群的最短距離大於每一個屬於該分群的查詢誤差容忍值的話，我們便可以將該分群的所有查詢 n-gram 忽略；反之，我們將該資料

n-gram 送至每一個相對應的查詢緩衝器（query buffer），以便於在組合答案的時候使用。

步驟三、合併查詢答案：合併機制（Merging Mechanism）的工作在於嘗試組合出能夠滿足某一查詢的近似答案。原則上，對於一個查詢來說，只要我們能夠找到滿足其每一個 查詢 n-gram 的資料 n-gram 時，我們就可以試著組合出一個可能的近似答案。然而如此鬆散的合併機制，將會浪費許多必不要的計算。因此，根據串流的特性以及誤差容許值的限制，我們在合成近似答案時採用了四項檢查，來避免組成一些不可能的答案，導致檢查時間的額外花費，這些檢查包括了滿足第 k 個查詢 n-gram 的資料 n-gram 必須在滿足第 k-1 個查詢 n-gram 的資料 n-gram 存在緩衝器中才屬有效；當組合出部分解答時，該部分答案的長度不能違反誤差容許值的限制；實際檢查目前資料 n-gram 與查詢 n-gram 的最小可能誤差，確定其沒有違反誤差容許值的限制；最後，在合成部分解答時，利用評估的方式預測已經產生的最小可能誤差，確定其沒有違反誤差容許值的限制。當我們完整組合出一段答案時，便要實際執行與查詢的距離計算，以確認是否為真正滿足之近似答案。

步驟四、實驗評估：我們所提供的實驗環境乃是利用不斷播放的音樂來形成一事件串流，每一個事件都代表了一個耳朵所聽到的複音，每一個複音都會由一個以上不同音高的音符所構成，此事件串流一共包含了 168283 個事件，播放的總時間為 26804 秒。為了能夠驗證本方法之效率，在實驗中所採用的比較對象乃是一個會對所有流入的資料片段進行距離運算與檢查之暴力法，圖六為我們其中一個具代表性的實驗成果。我們隨機選擇了長度為 24 個事件的片段作為查詢，我們可以看到當系統中註冊的查詢有 1000 筆時，每一首歌處理的時間與歌曲實際播放時間的比例，平均來說一首 3 分鐘的歌曲，我們只需要花費 29 秒鐘來處理，同樣的環境下如果使用暴力法，每一首 3 分鐘的歌曲我們必須要花費 3 分鐘 40 秒，所以本方法是可以實際滿足即時性需求的。



圖六：歌曲播放時間與處理時間之比例示意圖

四、未來工作

在未來一年的計畫執行中，表格資料之連續型查詢處理方面，會加入資料串流動態特性考量，發展可隨著資料串流自行調整執行規劃的連續型查詢處理技術；在串流監控技術方面，將開發資料串流上統計值的概算機制，用以監控查詢處理過程中流入與流出運算子的資料量，提供前一研究主題計算各運算子需要分配的系統資源。另外，對於序列資料，我們將延伸觸角至多屬性串流上的內容篩選技術，設計有效率的序列索引結構來儲存可能涵蓋不同屬性的查詢，並提出快速比對的演算法。

五、成果自評

本計畫為三年期計畫，在本年度的計畫執行過程中，我們已完成了原本預定於第一年完成的可擴充式連續型查詢處理、查詢串流之樣型探勘及多數值串流之內容篩選，研究成果包含相關研究論文五篇，一篇論文已公開發表於國際知名會議，兩篇論文已經投稿，另外兩篇論文為本年度碩士生之畢業論文。在第一年度的計畫完成之際，第二年度的計畫有關可調節式連續型查詢處理、資料串流之統計值概算、多屬性串流之內容篩選幾乎都已緊鑼密鼓的展開，透過本年度計畫的執行使得我們對於資料串流上的連續型查詢研究，累積了大量的相關研究經驗，對於往後二、三年的計畫執行將會有很大的幫助。

已發表之論文

[HWC06] Mark C. E. Hsieh, Y. H. Wu, Arbee L. P. Chen. Discovering Frequent Tree Patterns over Data Streams. 2006 SIAM Conference on Data Mining, Maryland, USA, April 2006.

已投稿之論文

[CWSA] H. C. Chen, Y. H. Wu, Steffi Soo and Arbee L. P. Chen. Continuous Query Processing over Event Streams Based on Approximate Matching Mechanisms. *Submitted for publication*.

[FC] Y. C. Fan and Arbee L. P. Chen. Exploring Common Evaluation for Efficient Processing Multiple Queries in Sensor Databases. *Submitted for publication*.

碩士生之畢業論文

Y. K. Lu. Mining Closed Frequent Subtrees over Data Streams, National Tsing Hua University, 2006.

Juliana Hsieh. An Efficient Optimization Strategy for Query Execution over Streaming Sources, National Tsing Hua University, 2006.

參考文獻

[AAKA02] T. Asai, K. Abe, S. Kawasoe, H. Arimura, H. Sakamoto, et al., "Online Algorithms for Mining Semi-structured Data Stream," *IEEE Conference on Data Mining*, 2002.

[AM04] A. Arasu and G.S. Manku, "Approximate Counts and Quantiles over Sliding Windows," *ACM Symposium on Principles of Database Systems*, 2004.

[AN04] A. M. Ayad and J. F. Naughton, "Static optimization of Conjunctive Queries with Sliding Windows over Infinite Streams," *ACM SIGMOD Conference*, 2004.

[BMMN04] S. Babu, R. Motwani, K. Munagala, I. Nishizawa, and J. Widom, "Adaptive Ordering of Pipelined Stream Filters," *ACM SIGMOD Conference*, 2004.

[CDTW00] J. Chen, D. J. DeWitt, F. Tian, and Y. Wang, "NiagaraCQ: A Scalable Continuous Query System for Internet Databases," *ACM SIGMOD Conference*, 2000.

[CE00] M. Clausen, R. Engelbrecht, et al., "PROMS: A Web-based Tool for Searching in Polyphonic Music," *International Symposium on Music Information Retrieval*, 2000.

[CF02] S. Chandrasekaran and M.J. Franklin, "Streaming Queries over Streaming Data," *VLDB Conference*, 2002.

[Dov99] M. J. Dovey, "An Algorithm for Locating Polyphonic Phrases within a Polyphonic Musical Piece," *AISB Symposium on Musical Creativity*, 1999.

[LP00] K. Lemstrom and S. Perttu, "SEMEX – An Efficient Music Retrieval Prototype," *International Symposium on Music Information Retrieval,* 2000.

[MF02] S. Madden and M.J. Franklin, "Fjording the Stream: An Architecture for Queries Over Streaming Sensor Data," *IEEE Conference on Data Engineering*, 2002.

[MM02] G. S. Manku and R. Motwani, "Approximate Frequency Counts over Data Streams," *VLDB Conference*, 2002.

[MSHR02] S. Madden, M. Shah, J. Hellstein, and V. Raman. "Continuously Adaptive Continuous Queries Over Streams," *ACM SIGMOD Conference*, 2002.

[VN02] S. Viglas and J.F. Naughton, "Rate-Based Query Optimization for Streaming Information Sources," *ACM SIGMOD Conference,* 2002.

[YCL04] J.X. Yu, Z. Chong, H. Lu, et al., "False Positive or False Negative: Mining Frequent Itemsets from High Speed Transactional Data Streams," *VLDB Conference*, 2004.

[YLH04] L. H. Yang, M. L. Lee, and W. Hsu, "Finding Hot Query Patterns over an XQuery Stream," *VLDB Journal Special Issue on Data Stream Processing*, 2004.