一個將關聯式資料庫轉換為圖形資料庫的方法*

Chin-Fu Lin[1], Sheng-Lung Peng[1], Ruay-Shiung Chang[2]
[1]National Dong Hwa University, Hualien, Taiwan
{810521001,slpeng}@ndhu.edu.tw
[2]National Taipei University of Business, Taipei, Taiwan
rschang@ntub.edu.tw

Abstract

關聯式資料庫一直是我們主要的資料存放系統，二個物件的關係可以透過鍵值的關聯來得到，如果要找相似度高的物件，必須要從所有可能鍵值查詢相關聯的物件，這是一個相當費事的工作。 於是有人提出圖形資料庫來解決這個問題[6, 11]，然而首要的工作是如何將關聯式資料庫轉為圖形資料庫，本篇論文提出了一個的方法，可以將一個關連式資料庫轉換成圖形式資料庫，在此基礎上，就能對其保有的資料做類似圖形的相似性搜尋。

1 緒論

由於網路技術的發展和普及，各種不同的線上服務已逐漸融入現代的生活之中， 尤其是各種網路社群平台，如今已是許多人不可割捨地日常的一部分。 在 Web 2.0 的概念成為主流之前，各種不同述求和面向的網站 (如：新聞網站、拍賣網站等等) 紛出， 而在今日，多數這些網站即便沒有轉型成為社群網站或被整併進其他社群網站， 也常常在非官方(甚至官方)的發言下，透過社群網路傳播訊息給更多的人知曉。

由於大量的使用者和多種不同面向的情報，有一定普及度的社群網站都面臨需要處理遠較傳統網站大上許多的資訊量的問題。 以 Facebook 為例，該網站光是新增資料的部分，每天都需要處理超過25億條的貼文和27億個 like， 更別說每天都還有大量的圖片、照片、影片等對系統負載更為吃重的多媒體資料被上傳至 Facebook。 保守估計，破100TB的資料都還只是每日新增資料的一般量， 可想而知這些社群網站的歷史資料將伴隨著服務的營運不斷膨漲，最終將累積至舊有的技術難以消化的量。 大數據正是為了因應這類問題而獨立出來的研究領域。

不論是在雲端大數據服務或傳統的網路服務中，資料(data content)永遠都是服務的主角。 為了方便資料的保存和管理，近乎所有含有動態資料的系統、服務都透過資料庫系統來管理各種資訊。 而在大數據的研究中，我們深切的體認到舊有的線性資料表已不敷使用， 而原本非主流的 NoSQL 和圖形資料庫也在這個背景下重新受到重視。

在資料探勘和大數據的研究中，如何在龐雜的資料中萃取出有價值的資料一直都是最重要的課題，這點在社群網路上當然也不會例外。 但現在很多社群網路的需求都無法被傳統的關聯式資料庫滿足，所以很多人(如：Rehman等人[9])都嚐試著設計能有效整合、分析大量網路資訊的資訊模型。 而這類模型最終的企圖也不外乎是想在海量的資訊中萃取出有價值的資料。 對於社群網路來說，非直觀又有價值的資料大致上可以分成兩大類：偵測發生的事件和發現潛在的需求。

我們可以將事件的偵測分為「偵測已知的事件」和「發掘未知的事件」兩種[1, 8]。 在偵測一個已知事件時，我們的目標是偵測有哪些使用者行為是和事件相關的。 舉例來說，在選舉前後，網路上常常會有比平常更多的關於宣傳和批判的發言或者是戲謔的改圖，而這些動作很有可能是因為選舉所引發的。 我們在偵測已知事件上，常常使用關鍵字、時間、地點、或帶有特定象徵的多媒體資料作為重要參數。 相關於同事件的行為雖然常常不會有統一的行為，但是通常大多數人的行為都會與少數幾個典型有高度的相似性(以球賽為例，典型往往是支持A隊或支持B球星)。

在發掘未知事件上，我們的目標是去發現是不是有一些我們不知道的事件發生，比如說A城市昨天晚上有一起火災。 通常對於發掘未知事件上使用的參數和用於偵測已知事件上的參數並沒有太大的區別(依然有時間、地點、關鍵字)[8]， 但因為我們根本不知道事件的發生，所以沒有辦法預先設定基礎關鍵字。 在此類問題上，我們判斷的目標依然是行為的相似性，而要求有兩項：和個體平時活動的相似性低、和群體當下的相似性高。 在達成此兩項條件的群體夠大時，應該就是彼時彼地有發生某些事件。如果該事件沒有和任何已知事件重疊的話，就有很高的機會是一個未知的事件。

*本計畫感謝科技部計畫編號：MOST 105-2221-E-141-002 之補助

而發現潛在的需求亦有兩個不同的層次：找出個人的需求[4, 7]和找出有共同需求的群體[3, 5]。找出個人的需求在實現各種社群網站的服務和大數據分析上有十分重要的意義。 現在社群網站上有機會流入每個使用者接受範圍內的資訊總量是十分龐大的，而且那些資料具有高度的多樣性，其多樣性不只在內容、主題、風格上，甚至連呈現方式的多樣性都十分地高。 在此背景之下，基本上沒有人能夠自行檢視所有可能收到的資訊，因此這項工作必需交給服務平台來完成。 在實作此項功能時，對於個人需求的分析和發現的成果好壞將直接影響到對於篩選給使用者的內容是不是能切中其所需。 但即便我們不將分析結果用在實作系統功能上，這項情報還是保有巨大的價值。 人在沒有其他因素的影響時，基本上會盡可能地施行滿足自己需求的行動(這也是為何我們會使用需求這個詞)， 所以如果我們可以準確預測個人需求的話，就可以在某種程度上預測使用者的行為。

找到有共同需求的群體也同樣有許多重要的應用。 群體的存在代表著在區域內有一定數量的使用者具有相似的行為，這就代表存在一個區域性的趨勢。 人是容易受環境和群眾影響的生物，所以如果可以誘導群體或潛在群體的需求，就相對容易誘導、產生屬於個人的需求。

不過上述所列的各種分析，不論是事件偵測還是需求發掘，全部都植基於相似性分析的基礎之上，而圖形資料庫無疑是目前最好的選擇。 本篇文論主要在提出一個方法，使得原本的關聯式資料庫可以很容易地轉換到圖形資料庫上，若將相似度轉換為圖論問題，那麼在轉換後的圖形資料庫上，就可以簡單、快速地得到需要的相似度查詢結果。

2 資料庫正規化

要在不影響資料相依性的前提下轉換資料庫，需要了解資料在關聯式資料庫中的結構。 為了容易保證資料的正確性和使用性，我們有時會需要改動資料表的設計， 為此而對資料庫做的調整即是正規化。 以下將列出比較常被使用的正規化規範。

2.1 第一正規化

在第一正規化下，於同一資料表中的任兩筆記錄都不會完全相同。

2.2 第二正規化

在第二正規化中，屬性被分為兩大類：Candidate key 和 Not key。 Candidate key 指的是可以特定出單筆記錄的最小參數組(所有非等於其自身的子集合都無法特定出單筆記錄)。

在第二正規化有兩條規範：

- 符合第一正規化
- 對所有的 Not key 參數 A 都存在至少一個 Candidate key C 能產生函數 $F : C \to A$。

2.3 第三正規化

第三正規化要求所有的 Not key 之間不存在任何的關連性。

假設在表 R 底下有一個函數 $F : X \to A$，X 是 R 的參數的子集，A 是 R 的其中一個參數，如果函數 F 符合以下其中一項條件，我們就說 F 符合第三正規化。

- $A \in X$。
- X 是個 Super key (存在一個 Candidate key C 可令 $C \subseteq X$)。
- 存在一個 Candidate key C 可令 $A \in C$。

如果一個資料庫中所有的函數都符合第三正規化的話，我們就可以確信該資料庫中所有的行為都和鍵值有關係。 換言之，不存在任何只介於 Not key 中間的操作。 也因為所有的函數都和 Candidate key 有關，所以如果符合第三正規化就一定符合第二正規化。

2.4 Boyce-Codd 正規化

Boyce-Codd 正規化是一個第三正規化的特例，他要求所有的函數 $F : X \to A$ 中，X 都必需是一個 Super key。 Boyce-Codd 正規化和第三正規化的差異在於，第三規化可能存在函數 $F : A \to B$，A 是 Not key 而 B 是 Candidate key 的一部分。

2.5 第四正規化

第四正規化是一個對 Boyce-Codd 正規化的一般化擴展。 在 Boyd-Codd 正規化中的函數 $F : X \to A$ 中，A 是資料表 R 的一個參數。 在第四正規化中的函數 $F : X \to Y$，Y 可以是一個參數組。

在第四正規化下，每個函數 $F : X \to Y$，Y 都符合下列兩點中的至少一點：

- X 是個 Super key。
- $X \cup Y = R$。

2.6 第五正規化

在關聯式資料庫中，我們有很多搜尋結果是靠 join 多個表得到的， 有些符合第四正規化的表格可能可以分割成多個小表格，利用 join 來得到原本的關連。 為了確定我們是不是有可能利用多個表來 join 出關連 R, 研究者們定義了 join dependency。

當有一個關連 $R = join\{A_1, A_2, \ldots, A_n\}$ 且所有 $A_i \subseteq R, 1 \leq i \leq n$。 (join 項目不一定要來自同一個表。) 如果只在所有 A_i 都是 R 的 Super key 的情況下 R 才能被特定的話，我們就說此 join function 符合 join dependency。 而當所有的操作都符合 join dependency 時，此設計就符合第五正規化。

當一個資料庫的設計符合第五正規化時，代表這個資料庫裡面所有的表都沒辦法在不損失任何關連的前提下把表切得更小[2]。

2.7 正規化對效率的影響

除保證資料的正確性和一致性之外，資料庫正規化還有去除資料庫中冗餘部分的功效，有助於提升資料儲存和運算的效益。 簡略來說，普遍認為符合正規化的資料庫在運算和記憶空間效率的表現上都優於沒有經過正規化的資料庫； 正規化的層級越高對空間效率的表現愈好； 而正規化層級越低則對關聯性運算的表現愈佳。 Ting J. Wang 等人的研究則給了我們更詳細的結果[10]：

- 第一正規化提升了維護、查詢單一資料表時的效能。
- 第二正規化大幅提升了空間效率(和第一正規化比較，僅佔用約 20% 的空間)和更動資料(新增、修改、刪除)的執行效率。
- 第三正規化小幅提升了空間效率(和第二正規化比較，佔用約 96% 的空間)和更動高獨立性資料的執行效率(在各資料相依性複雜的情況下，效率不如第二正規化)。

市面上大多數的關聯式資料庫都以線性資料表做為主要儲存格式。 在該類資料庫中，每個帶條件的簡單搜尋都需要花上正比於資料表大小的時間， 在資料量極龐大的前提下，線性時間的搜尋效率已不再有辦法於可接受的回應時間和等待時間(response time, waiting time)內完成。 相對於此，圖形式資料庫的搜尋可以以某些節點為根節點，只通過指定的關聯類型掃視附近的點， 不需要每次都搜尋整個資料庫，或者可以在多個足夠短的等待時間內分批傳回多組搜尋結果(而且通常優先度越高的越早被找到)。

在圖形資料庫的搜尋過程中，時間不是花在節點上的資料比較就是花在關聯(Relation, Edge)的掃描， 假定我們需要掃描資料庫中的一個圖形 $G = (V, E)$ 上所有的節點，則我們所需耗的時間將是 $\sum_{v \in V} O(L(v)) + \sum_{e \in E} O(L(e))$，$L(x)$ 代表 x 上的資料(labels)。 由此可知在圖形資料庫中的資料搜尋與操作所需的時間約正比於總資料量。 也因此我們可以相信，在使用圖形資料庫時我們不需要太擔心正規化程度過高帶來的運算負擔。 而由於第五正規化保證了資料表最小化，所以本論文的方法針對符合第五正規化的關聯式資料庫來做轉換。

3 資料庫的轉換

一個資料庫的功用通常是由使用者對資料庫會做的行為來決定。 在前節資料庫正規化的說明中，我們將所有會對資料庫做的操作都以函數定義。 只要可以實行滿足所有定義的操作函數，不論物理層面實質上對資料做了怎麼樣的操作， 在高度抽象的層面上都可以視為等價的行為，也不會有任何預期外的資料漏失。 在所有的操作都有等價的操作可以用，資料也沒有任何的漏失的話，我們就可以說這兩個資料庫等價。 所以我們可以將轉換資料庫形式的行為分成兩個部分：設計可無損乘載所有原資料的資訊模型和實作所有會對原本資料庫做的操作的等價操作於新的資料模型上。

對於一個傳統的關聯式資料庫來說，資料都是由線性表格所組成； 每個表格都有一個以上的欄位，每個欄位用來儲存一個參數； 每個表格都可以有零組以上的記錄，每組都是一個含有該表全參數的元組(Tuple)，但可能允許空(Null)值。 而關於操作的部分，主要是以 SQL 語言來操作。 SQL 對資料庫操作的支援可以完全滿足第五正規化所會做的操作(join query)。

圖形式資料庫是則由節點(Vertex)和邊(Edge)兩種元件所組成。 一個節點可以接續零個以上的邊，而自身亦可以保有資料(Label)。 一個邊同時接續著兩個節點，分別代表起點和終點，其自身亦可以保有資料(Label)。 一個無向邊的兩端點都可以任意視為起點或終點，只是起點和終點必需成對出現(不能同時是起點或同時是終點)。

對於關聯式資料庫轉換至圖形資料庫上我們需要注意兩點：

- 原資料庫中的所有記錄的全部欄位的值都會出現在某個節點或邊的 Label 上。
- 在原資料庫的操作中可以找得到的關連在圖形中都存在相對應的尋訪路徑。

我們提出的轉換方法適用於符合第五正規化的資料庫。 在第五正規化中所有的操作都是符合 join dependency 的， 而我們可以將表格分成兩種：

- 所有欄位的資料都可以透過別的資料表 join 出來。
- 含有無法被 join 出來的資料欄位。

對於任何含有無法被 join 出來的欄位的表格，代表其一定含有獨立於全部關聯的純資料。 我們將這類表格的每一筆記錄都獨立成一個節點，並將所有欄位上記錄的資料標記於其 label 上 (見 Table 1, Figure 1)。

id	1
account	account_a
name	User Name

Table 1: 表 *users* 中的一筆資料

Figure 1: 該資料轉換為節點

對於資料欄位含有 foreign key 的欄位，建立連線至所關連的資料點，並將對應的 key 記錄於該連線之 label 上 (見 Table 2, Figure 2)。

club	Tennis
owner	1

Table 2: 表 *clubs* 中的一筆資料

對於所有欄位都可以由他表 join 出來的表而言，其一定符合 $R = join\{A_1, A_2, \ldots, A_n\}$ 的形式， 且所有的 A_i 一定有其來源表格。 我們可以建一個不帶任何 label 的空節點，並用標記著表格名的邊連接 A_i 的來源節點。 對於一個連接著兩個帶著同標記的邊的空節點($R = join\{A_1, A_2\}$)，可以用單一個邊來取代 (見 Table 3, Figure 3, 4)。

我們亦可以將資料庫中合法的操作分成兩種：

- 符合第四正規化。
- 需要使用 join 操作。

在第四正規化所構想的操作中，只有兩種：

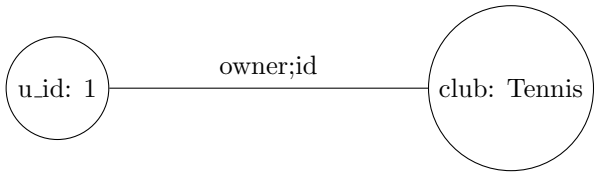

Figure 2: 以 foreign key 來建立連接

user_id	1
club_id	3

Table 3: 表 *members* 中的一筆資料

- 所有資料都在同一表內(Trivial)
- 透過包含他表 Superkey 的 Foreign key 來建立鏈結(Non trivial)

因為該轉換並沒有省略任何原表格上的資料，所以我們可以確定， 當我們於轉換的圖上進行一個 trivial 操作時， 其所有的欄位資料必定可以在該節點(data field)或該節點的鄰居(foreign key)上找到。

因為所有的 foreign key 都會被轉換為連接邊連至相關節點，所以所有的 non trivial 操作在轉換的圖上一定存在一條 path。 path 存在資料庫中就意謂著我們可以進行等價的操作。

關於需要使用 join 操作的部分，因 join 是根據 foreign super key 來決定要 join 哪些 record。 我們已知所有的 foreign key query 都存在 path，foreign super key 也是 foreign key 的一種， 所以對於所有可被 join 出來的表，我們必定找得對與其相對應的 connected subgraph。

因此，在符合第五正規化的前提下，我們可以很直觀地看出所有原本會做的操作一定存在等價操作於本方法產生的圖形中。

4 結論

如果原本的資料庫設計合乎 domain-key 正規化的規範的話， 將會有許多同值的點可以被濃縮為單一個節點，可以有效節省記憶空間的耗用。但即使不合乎 domain-key 正規化的規範，只要合乎第五正規化，本文提出的方法還是可以正常作用。

本論文中提出的 Migration 方法只保證符合第五正規化的資料庫， 但應該要能接受第三正規化才更能切合實務上的需求，但第三正規化的轉換過程系統化還有待研究。

未來我們可以試著將同類的節點或者同分群的節點 decomposition 成一個 super vertex。 而這

Figure 3: 該關連表轉換為節點

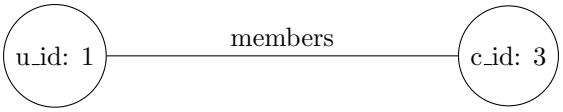

Figure 4: 該二關連節點轉換為連接

些 super vertices 將會構成一張新的圖 G'。 我們可以利用圖 G' 來探討各群體之間的關係。

References

[1] B. Becker, D. Iter, M. Naaman, L. Gravano, "Identifying Content for Planned Events Accross Social Media Sites," *WSDM*'12, pp. 533–542, 2012.

[2] H. Darwen, C.J. Date, R. Fagin, "A Normal Form for Preventing Redundant Tuples in Relational Databases," *ICDT*'12, pp. 114–126, 2012.

[3] A. Goel, A. Sharma, D. Wang, Z. Yin, "Discovering Similar Users on Twitter," *Proceedings of the Workshop on Mining and Learning with Graphs*, 2013.

[4] M.A. Hasan, M.J. Zaki, "A Survey of Link Prediction in Social Networks," *Social Network Data Analytics*, pp. 243–275, 2011.

[5] H. Kwak, H.-Y. Shin, J.-I. Yoon, S.B. Moon, "Connecting Users with Similar Interests Across Multiple Web Services," *Proceedings of the Third International Conference on Weblogs and Social Media*, 2009.

[6] S. Jayaraman, S. Viswanathan, "Comparative Survey of Query Processing on Graph Databases," *COP*5725, 2013.

[7] D. Liben-Nowell, J.M. Kleinberg, "The Link-Prediction Problem for Social Networks," *Proceedings of the 2003 ACM CIKM International Conference on Information and Knowledge Management*, pp. 556–559, 2003.

[8] F. Psallidas, H. Becker, M. Naaman, L. Gravano, "Effective Event Identification in Social Media," *IEEE Data Eng. Bull.*, Vol. 36, No. 3, pp. 42–50, 2013.

[9] N.U. Rehman, S. Mansmann, A. Weiler, M.H. Scholl, "Building a Data Warehouse for Twitter Stream Exploration," *ASONAM*'12, pp. 1341–1348, 2012.

[10] T.J. Wang, H. Du, C.M. Lehmann, "Accounting for the Benefits of Database Normalization," *American Journal of Business Education*, Vol.3, No. 1, pp. 42–52, 2010.

[11] X. Yan, P.S. Yu, J. Han, "Substructure Similarity Search in Graph Databases," *SIGMOD*'05, pp. 766–777, 2005.