科技部資訊安全技術研發專案計畫

『系統測試計畫書』

System Testing Plan Document

連續性時序資料的隱私保護機制-以巨量個人定位資料為例

Privacy Protection Mechanisms for Continuous Time Sequential Data - A Case Study Based on Big Personal Location Data

MOST 105-2221-E-415-016-

研究團隊

主持人：林土量助理教授(國立嘉義大學資管系)

Department of Information Management,

National Chiayi University, Taiwan

2017/06/14

目錄

版次變更記錄

Github 網址

(https://github.com/lozzr1012/Tu-Liang-105-08-106-07-Project/)

版次	變更景點	變更日期
1.0	實作頻繁景點探勘	2017/1/11
2.0	修改頻繁景點探勘 Bugs	2017/2/8
3.0	修改不頻繁景點 Bugs	2017/3/13
4.0	完成不頻繁景點探勘	2017/4/4
5.0	完成定位資料的不頻繁景點探勘與泛化	2017/6/13

1. 緒論（Introduction）

在這個網路技術快速發展的年代,使用者可透過電腦和行動裝置享受網路上各式各樣的服務，如:線上購物、搜尋引擎、網頁瀏覽…等，在這同時網路服務供應商也在收集使用者的相關記錄，並期望能從這些使用記錄資料中找出有價值的資訊，這些使用記錄通常和時間有關而被稱為連續性時間序列資料，透過資料探勘這些連續性時序資料，雖然可以讓企業更了解客戶的行為，進而改善其商業模式或服務品質。

相較於靜態資料，連續性時序資料在隱私保護上相對複雜，因此如何設計一可兼顧時序資料隱私保護和資料可用性的演算法有其實務和學術上的重要性。傳統常用的隱私保護的措施有k-匿名法(k-Anonymity)[1]和差分隱私保護(Differential Privacy)[2]，k-匿名化或差分隱私保護技術都希望能達成合理的隱私資料保護，且同時不破壞資料的可用性。不過這些傳統匿名技術通常都假設需要匿名化的資料是靜態的，並未考慮到資料的動態新增，但許多的資料都屬動態的連續性時間序列資料(Time Sequential Data)，如：消費記錄、看診資料、GPS 軌跡、網頁瀏覽記錄等,如何達成連續性時間序列資料的隱私保護是當前資訊安全領域中非常重要的一項研究議題。目前 k-匿名技術並未考量連續性時間序列資料的隱私保護，以王小明的例子來說，或許獨自出現在台北車站或台北 101 的靜態資料已經 k-匿名保護，但台北車站→台北 101 的序列資料並未匿名保護，因此惡意攻擊者還是可以用序列資料來進行隱私的挖掘。

目前最廣泛被使用在連續性序列資料探索的兩項技術是：關聯規則(Association Rule)擷取[3]及序列比對(Sequence Alignment)[4]，關聯規則擷取主要是用於發現資料間的隱藏關係，透過資料景點不同粒度等級（Granularity Levels）的結合產生多個分類群集，來找出隱藏關係。

本計畫主要開發匿名化連續性時序資料,避免個人隱私資料被惡意攻擊者利用序列資料探勘技術識別出來。匿名技術與資料探勘技術有其相關性，尤其連續性時間序列資料所隱含的知識通常都需要透過序列資料探勘技術來取得,因此在開發連續性序列資料的隱私保護機制時，必須把序列資料的探勘技術列入考量。所以本計畫將基於目前最廣泛使用的關聯規則探勘技術來開發匿名機制，並將其應用在巨量個人定位資料。

雲端的計算近年來受到極大的重視。許多的科學計算常需要巨額的軟硬體設備投資，因而讓研究人員望而怯步，雲端計算的發展可紓解研究人員在這方面的困擾，讓研究人員可以直接使用雲端計算資源供應商所提供之高效能與高穩定度的軟硬體平台，來快速獲取複雜科學計算的結果，以達成之前所未有的效率[5]。因此雲端的技術吸引了具有大量資料及高度的計算需求的應用，巨量連續時序資料的隱私保護具有會使用到大量資料和高度計算需求的特性，所以是很適合開發成雲端的應用，配合雲端虛擬主機的配置，可縮短隱私保護程式的運算時間，進而擴展應用到更寬廣的範圍。雲端部分是透過 MapReduce 運算架構，是由 Google 所提出的 MapReduce 軟體架構，一分散式與平行化處理的程式設計模型，可以同時運行在多個不同電腦組成的叢集上，它主要是用來處理大量資料，擁有可靠的容錯機制。不同於典型的設計架構，主要由 Map 與 Reduce 兩個函式組成負責整個軟體架構的工作。MapReduce[6, 7]的程式架構特性，能加快產生頻繁景點集的速度，圖 1 為 MapReduce 的實例資料流程圖。

1. Map 階段：該函式所負責的任務是從主節點(master node)輸入的一個 key/value 序對，將這組輸入分割成數個的子部分，分散到各個工作節點(worker nodes)上做運算。

2. Reduce 階段：該函式負責由主節點(master node)收回處理完的子部分，將子部分重新組合產生輸出。

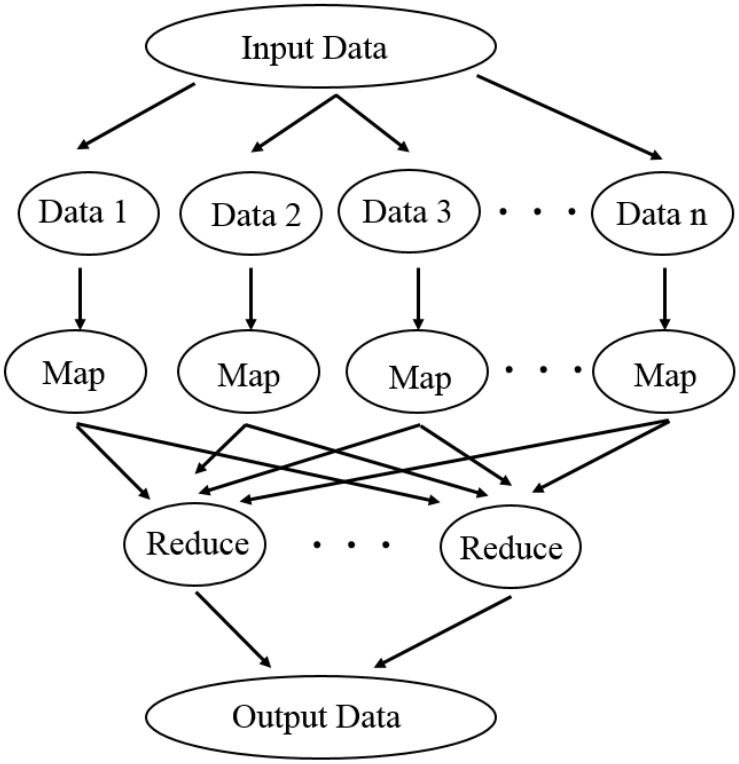

圖 1.MapReduce 實例資料流程圖

我們團隊針對了 MapReduce 以及頻繁景點探勘 FP-Growth 做結合，則我們方法分成了兩個步驟。

1. 首先，先針對資料庫做景點的頻繁次數統計，針對了每一筆打卡景點做計算，利用 MapReduce 架構下，可以將各個景點做分散式的計算，並且針對景點做降冪排序。
2. 接著，這個步驟算是本團隊演算法整個核心，透過上一步驟可以得到景點的支持度，將頻繁景點利用了 MapReduce 架構，把 FP-Tree 挖掘分散到各個節點做建立，接著針對各個輸出的結果做合併。最終可以得到頻繁景點的關聯規則。

保護不頻繁景點則是透過泛化方式呈現，若從原始資料庫中透過資料探勘可以探勘出頻繁景點及不頻繁景點。假設透過探勘步驟可以得到頻繁景點為 EA：5 次，在不頻繁景點中探勘出 EC：1 次，我們團隊會結合頻繁景點和不頻繁景點，透過泛化的方式讓 EA：5 和 EC：1 轉換成 EX：6。因此在不頻繁景點中，就不會單獨有一筆 EC 資料存在，透過泛化方式組合出 EX：6，

達到資料的隱私保護，如圖 2。

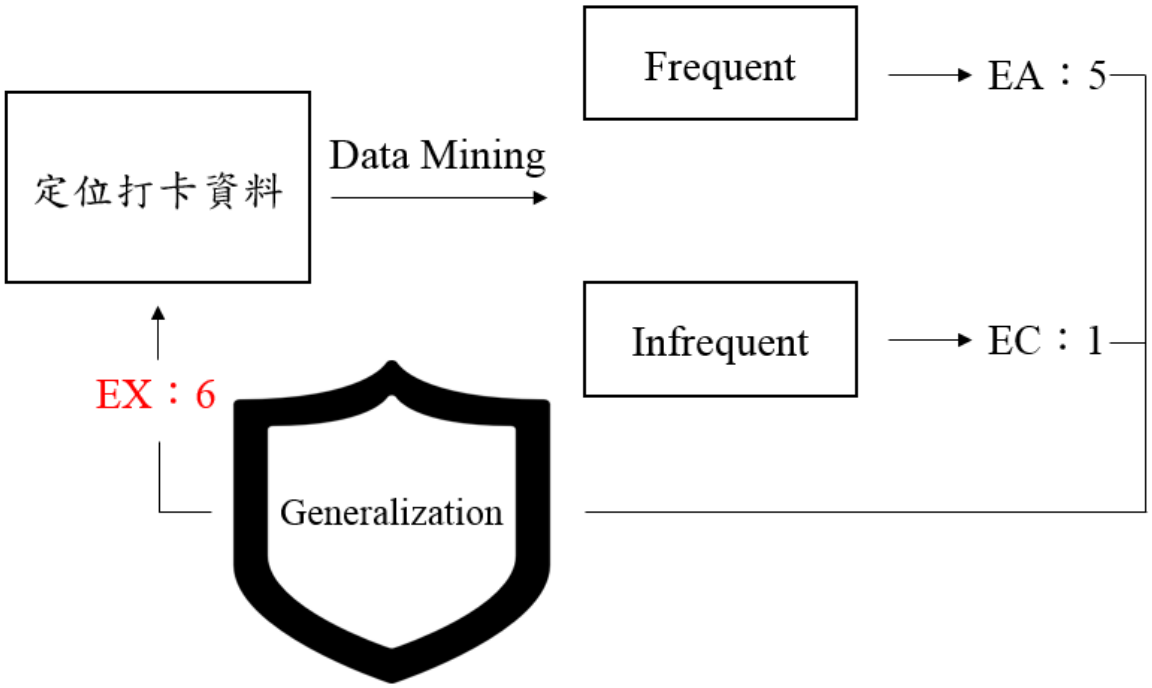

圖 2. 保護不頻繁景點集示意圖

本計畫主要探討如何匿名化連續性時序資料，避免個人隱私資料被惡意攻擊者利用序列資料探勘技術識別出來。所以本計畫主要是透過 FP-Growth 找出不頻繁景點集，並且利用雲端 MapReduce 架構下進行。則探勘出的不頻繁景點利用泛化(Generalization)方式保護較少出現的資料，達到匿名的效果，並建構保護不頻繁景點的系統。依據本系統可進行各子模組整合測試和接受測試(Acceptance Testing)：

(1) 景點統計次數模組(Spot Counts of Statistics Module, ACSM)

(2) 資料儲存模組(Data Storage Module, DSM)

(3) 不頻繁景點模組(Infrequent Spot Module, IPM)

(4) 保護不頻繁景點模組(Protect Infrequent Spot Module, PIPM)

1.1 測試範圍 (Scope of Testing)

對於連續性時序資料的隱私保護系統的測試範圍，將包含單元測試、整合測試、以及接受度測試。首先會先進行單元測試，程式開發者於系統設計過程中，會不斷的進行各種函式的認證以及成果的探討，並確保單一功能正常運作。接著單元測試後緊接著整合測試，整合測試的過程中，會去驗證像是景點統計模組是否能順利的利用 FP-Growth 做探勘，使系統能順利的探勘出不頻繁景點集。最後則會讓使用者進行系統測試以及接受度測試，查看使用者會不會面臨到功能性的問題。

1.2 接受準則 (Acceptance Criteria)

本測試計畫需要滿足下列的測試接受準則：

(1) 景點統計模組必須能夠順利讀取到系統產生的定位資料。

(2) 景點統計模組必須能夠對資料做次數的計算統計。

(3) 資料儲存模組必須能夠針對景點統計產生出來的資料做儲存。

(4) 不頻繁景點模組能夠順利讀取到資料儲存的景點統計次數。

(5) 不頻繁景點模組必須能夠從根據景點統計模組探勘出不頻繁景點集。

(6) 保護不頻繁景點模組能夠順利保護較少出現次數的做泛化。

(7) 各項參數與數據必須能夠在資料中正常的傳遞。

(8) 測試程序必須依照本測試計畫所制定的程序來進行，且測試結果必須能夠符合系統需求規格書預期測試結果方能接受。

2. 測試環境 (Testing Environment)

連續性時序資料的隱私保護系統測試環境架構圖描述如圖 3：

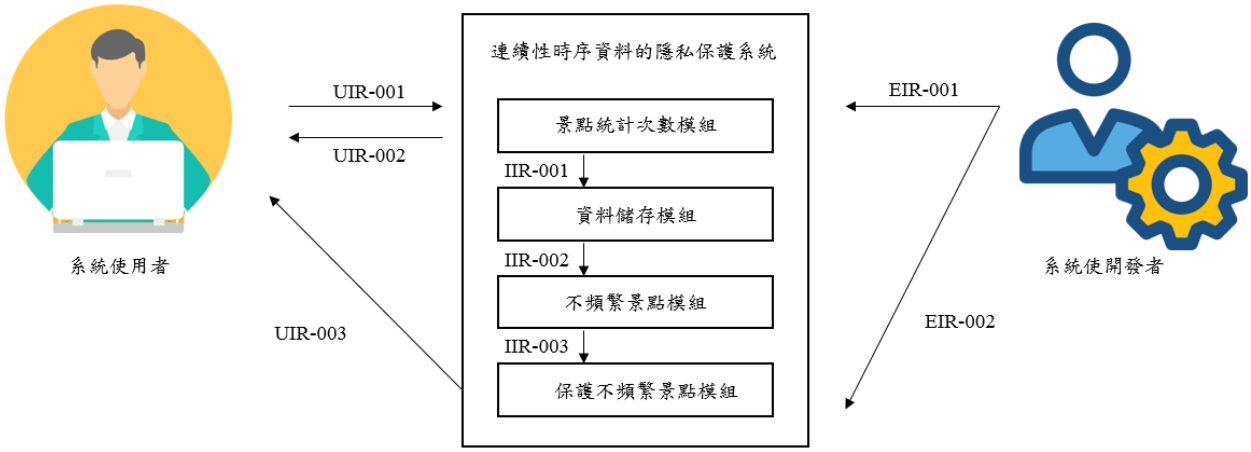

圖 3.連續性時序資料的隱私保護系統測試環境架構圖

2.1 硬體規格 (Hardware Specification)

依據圖 3 測試環境架構圖內容，進行測試之硬體規格說明，如下列所示：

系統使用者

CPU：Intel(R) Core(TM) i5-4570 CPU @ 3.20GHz

RAM：8.00GB

HDD：1TB

2.2 軟體規格 (Software Specification)

依據圖 3 測試環境架構圖內容，測試環境之轉體規格說明，如下列所示：

連續性時序資料的隱私保護系統：包含景點統計模組、資料儲存模組、不頻繁景點模組、保護不頻繁景點模組

作業系統：Windows 7 Ultimate Version 64 bits

系統開發平台：Eclipse

系統開發語言：Java

雲端環境：windoop_1.0.3_app_mode_zh_TW

FP-Growth(Frequent Pattern Growth) 演算法[8]改善 Apriori 演算法的記憶空間需求，該演算法不需產生候選景點集，其中的 FP-Tree 資料結構能有效的壓縮資料庫。針對資料庫中的資料組合作最小支持度門檻值的篩選並建樹，以迭代的方式反覆對 FP-Tree 的路徑進行條件比對，並探勘出所有的頻繁景點集。Apriori 演算法將資料存放在硬碟當中，在每回合判斷候選景點集是否為頻繁景點集時，都需要重新掃描資料，因此會增加運算時間。FP-Growth 演算法則是利用 FP-Tree 資料結構將資料庫做壓縮，並將壓縮後的資料存放記憶體中，因此相較於 Apriori 演算法，在處理過程中掃描硬碟資料的次數僅只需兩次，且運算速度相對加快許多。

FP-Growth 演算法分成了兩個部分，第一部分是建樹，第二部分是探勘，在探勘頻繁景點集時，將 FP-Tree 產生的樹狀結構利用遞迴方法反覆對整棵樹進行比對，探勘出所有頻繁景點集。

(1) 建立 FP-Tree

第一步驟：掃描資料庫，利用資料庫中的資料，透過景點出現次數探勘出大於或小於等於門檻值的頻繁景點，並依照景點(Item)出現次數做降冪排序建立頭表(Headtable)，表 1。

第二步驟：再次掃描資料庫，並根據資料庫的景點(Item)建立 FP-Tree。

(2) 探勘 FP-Tree

第一步驟:針對每一個 FP-Tree 的分支節點，建立條件模式(Conditional Pattern Base)。

第二步驟：接著針對每個條件模式(Conditional Pattern Base)分別建立 Conditional FP-Tree。

第三步驟：針對條件的 FP-Tree 葉節點(Leaf)至根部(Root)以遞迴方式進行從底部往上(Bottom-Up)探勘。

第四步驟:Conditional FP-Tree 包含一個完整的關聯路徑，即可列出所有模式(Pattern)，如圖 4。

表 1.原始打卡記錄

TID	Item	Ordered Frequent Items
1	d,a,c,f,g,i,m,p,t	f,c,a,m,p
2	a,b,o,f,l,m,c	f,c,a,b,m
3	b,f,h,o, j,v	f,b
4	b,p,k,s,c,v	c,b,p
5	c,f,a,e,l,m,p,n,r	f,c,a,m,p

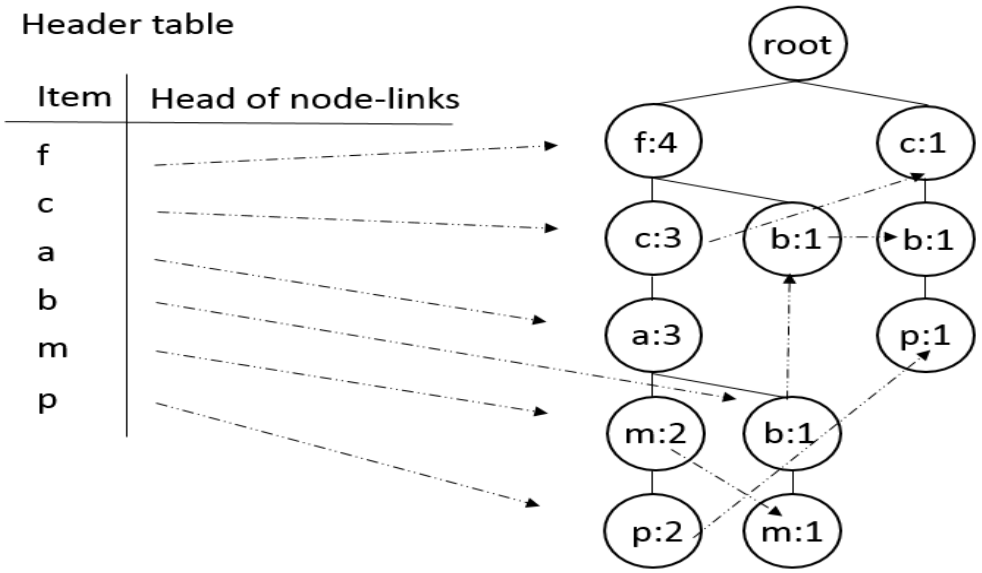

圖 4.FP-Growth 演算法實例推導圖

2.3 測試資料來源 (Test Data Sources)

測試景點	數量	測試來源
隨機產生景點資料	50 筆	為測試本系統，本團隊隨機產生打卡的景點作記錄。
隨機產生景點資料	100 筆	為測試本系統，本團隊隨機產生打卡的景點作記錄。
隨機產生景點資料	150 筆	為測試本系統，本團隊隨機產生打卡的景點作記錄。
隨機產生景點資料	200 筆	為測試本系統，本團隊隨機產生打卡的景點作記錄。
NYC Restaurant Rich Dataset	2000 筆	https://sites.google.com/site/yangdingqi/home/foursquare-dataset

3. 測試時程、程序與責任 (Testing Schedule, Procedure, Responsibility)

3.1 測試時程 (Testing Schedule)

(1) 時程

測試景點	時間
字母統計次數模組單元測試 (Unit Test)	2017/02/10~2017/02/15
資料儲存模組單元測試 (Unit Test)	2017/02/16~2017/02/20
不頻繁景點模組單元測試 (Unit Test)	2017/02/21~2017/02/29
保護不頻繁景點模組單元測試 (Unit Test)	2017/03/1~2017/03/10
各模組之整合測試 (Integration Test)	2017/03/11~2017/03/31
各模組之接受度測試 (Acceptance Test)	2017/04/1~2017/04/30

(2) 查核

測試景點	時間
各模組之單元測試 (Unit Test)	2017/05/01~2017/05/15
各模組之整合測試 (Integration Test)	2017/05/16~2017/05/31
各模組之接受度測試 (Acceptance Test)	2017/06/01~2017/06/15

3.2 測試程序 (Testing Procedure)

單元測試及整合測試的步驟是程式開發人員修正錯誤,並且由相關模組程式開發的人員進行檢核。等待各個模組整合結束後,由程式開發人員進行接受測試,並且由程式開發人員進行查核及移除錯誤。

3.2.1 整合測試(Integration Testing)

程式開發人員透過了產生打卡景點的系統進行整合測試，測試包含了所有模組(景點統計次數模組、資料儲存模組、不頻繁景點模組以及保護不頻繁景點模組)，並以整合這些模組為目的。整個系統的使用案例如下，系統使用者透過景點個別出現次數做統計，將資料儲存做存取，並且利用 FP-Growth 來探勘不頻繁景點集。最後再透過了泛化的方式來保護探勘出來不頻繁景點集的資料。

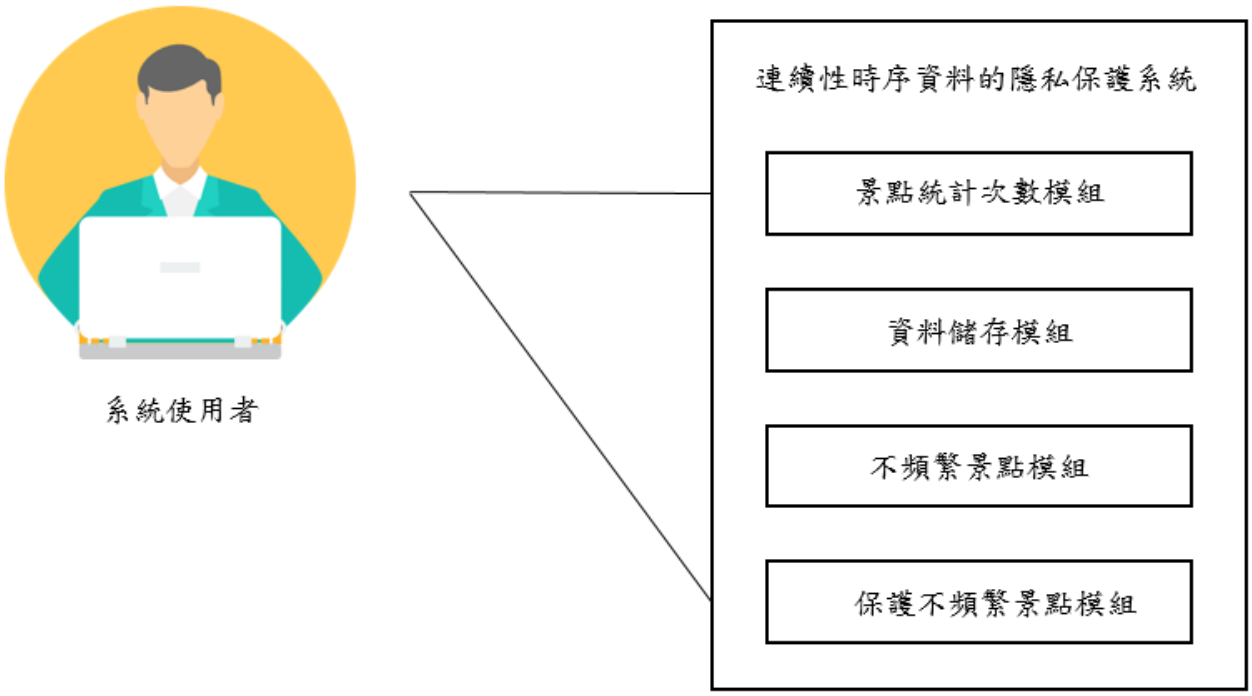

圖 5.連續性時序資料的隱私保護系統使用範例

景點統計次數模組與資料儲存模組測試如下所示，主要測試是使用者針對打卡資訊的景點統計，計算出各景點的出現次數，並且輸出景點出現次數檔案。並且把輸出的檔案做資料儲存模組，能到下一步驟也能輸入檔案。

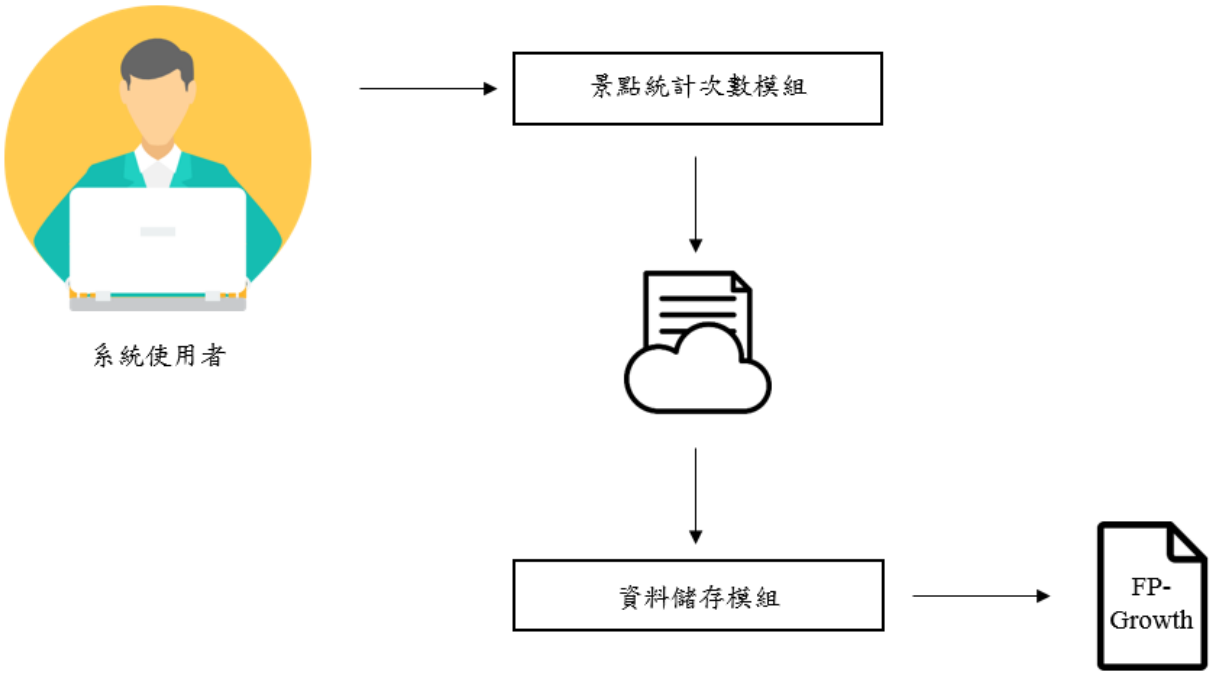

圖 6. 景點統計次數模組與資料儲存模組之整合測試(IT1)

透過上一個階段輸出資料儲存模組的景點統計，利用的 FP-Growth 演算法讀取檔案景點次數資料做探勘，並且探勘出所有小於門檻值的景點，則稱不頻繁景點集。所產生出來的資料則稱待受保護資料，這些資料屬於不頻繁景點也代表在打卡記錄較少出現，該階段找出較敏感資料，不過尚未針對這些資料做保護。

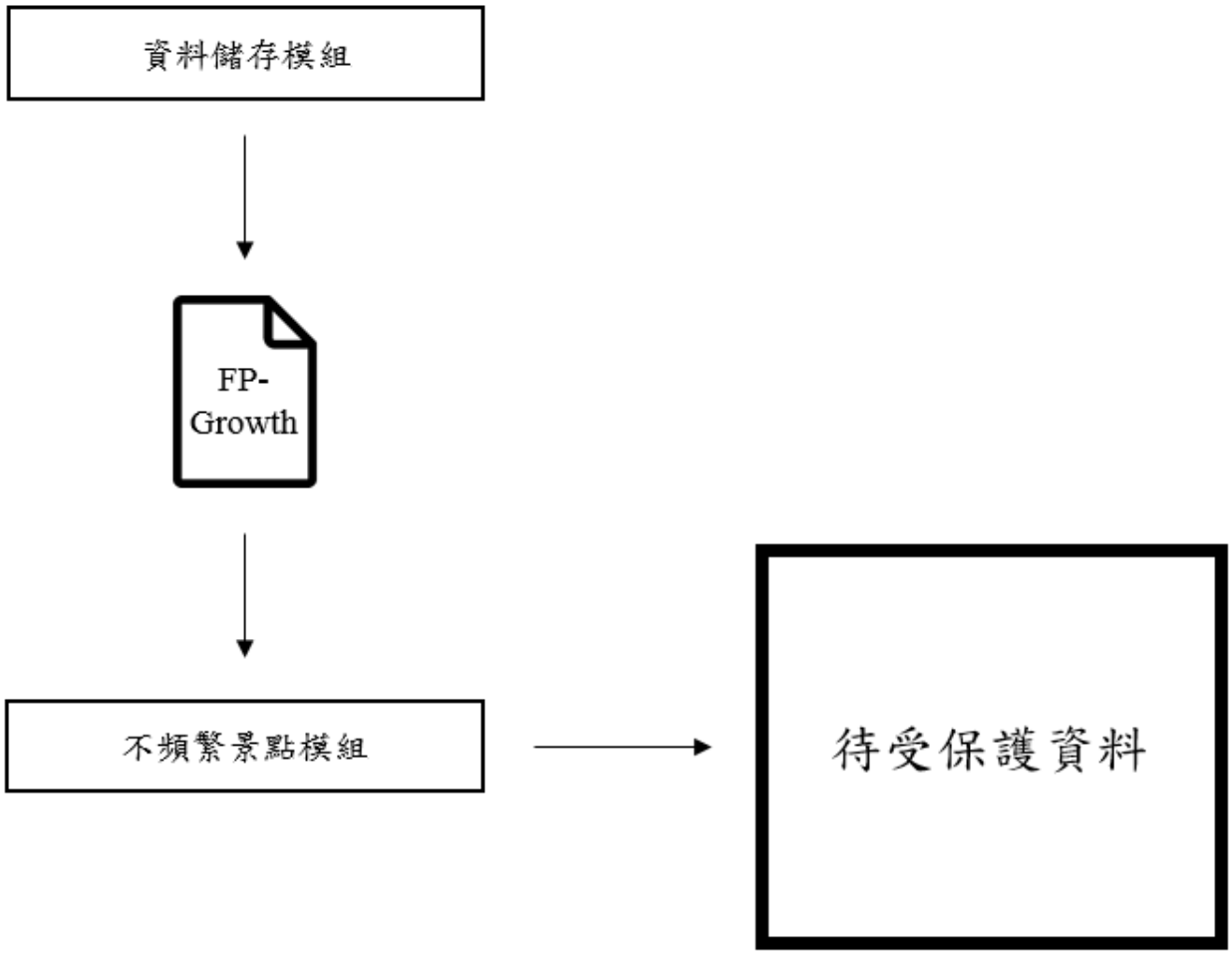

圖 7. 資料儲存模組與不頻繁景點模組之整合測試(IT2)

最後是針對探勘所產生的待受保護資料作處理，透過了不頻繁景點找到這些較敏感的資料，我們必須針對這些在原始資料庫較少出現的資料作保護。利用了泛化的方式做處理，從待受保護打卡資料尋找相似的關聯，讓原本低於門檻值的不頻繁景點成為頻繁景點，這樣就能達到保護不頻繁景點模組，並且同時也要確保資料可用性，最終達到匿名保護效果同時也兼顧資料特性。

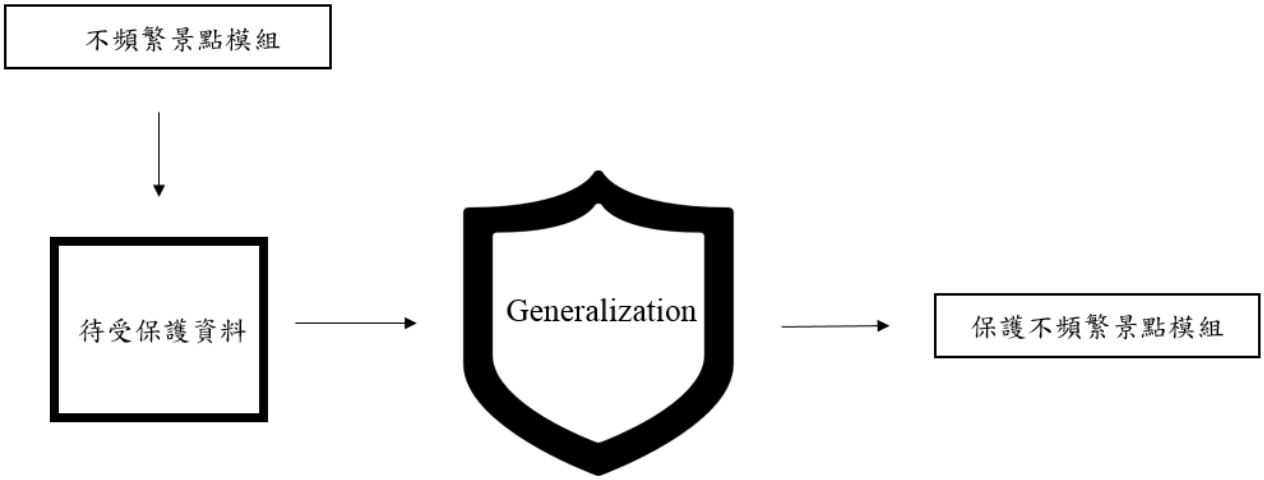

圖 8. 不頻繁景點模組與保護不頻繁景點模組之整合測試(IT3)

3.3 人員職責分配 (Personnel Responsibility Assignment)

表 2.人員職責分配表

測試項目	人員
Unit Test	溫柏為
Acceptance Testing (AT1)	溫柏為
Acceptance Testing (AT2)	溫柏為
Integration Testing (IT1)	王品傑
Integration Testing (IT2)	王品傑
Integration Testing (IT3)	王品傑

3.3.1 接受測試（Acceptance Testing）

接受測試主要分成初步使用以及後續保護的部分。初步使用主要是透過景點統計次數模組、資料儲存模組、不頻繁景點模組進行測試。則初步使用該系統必須透過產生景點的打卡記錄做探勘，尋找出在打卡中的不頻繁景點的資料。

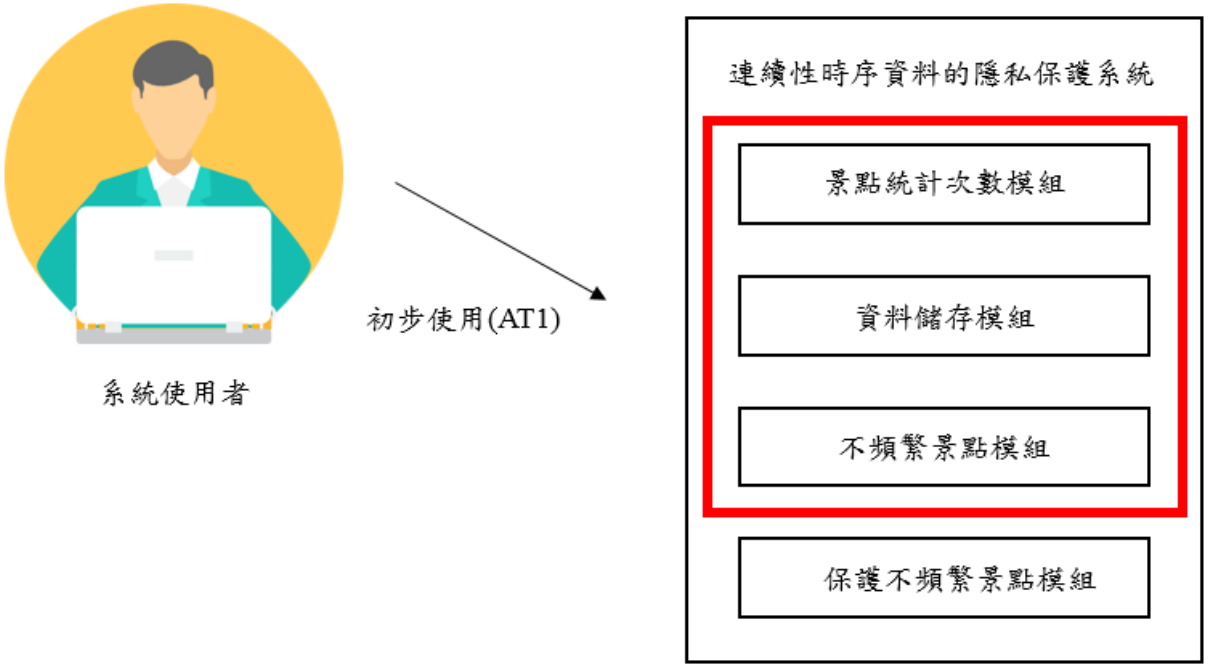

圖 9.初步使用之接受度測試(AT1)

接著進行探勘出來的不頻繁景點做處理，主要是透過保護不頻繁景點模組進行接收度測試。當系統找出不頻繁景點，透過了泛化的方式進行保護。當系統找出這些景點低於門檻值的景點，讓不頻繁景點變成頻繁景點，讓資料受到一定的保護，並且保有資料本身的特性。

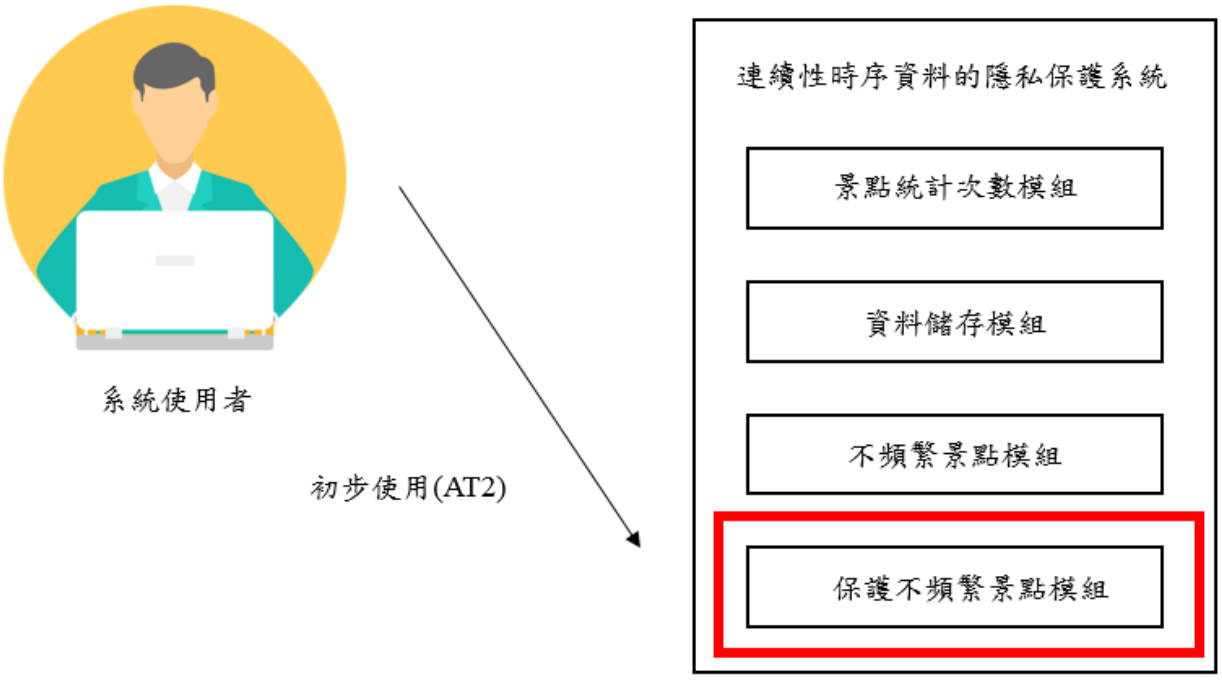

圖 10. 後續保護之接受度測試(AT2)

4. 測試案例 (Test Cases)

4.1 整合測試案例 (Integration Testing Cases)

4.1.1 IT1 測試案例

目的：

(1) 驗證景點統計模組是否正確統計出打卡資料的次數。

(2) 驗證景點統計模組產生的統計次數格式完整。

表 3. IT1 測試案例

Identification	IT1	
Name	整合景點統計次數模組與資料儲存模組	
Tested target		
Reference		
Severity		
Instructions	*Actor actions*	*System responses*
	1. 景點打卡資料。	
		2. 透過產生的打卡資料做景點統計，記錄打卡資料中的景點做次數計算。
		3. 透過產生的景點次數統計，確認格式無誤。
Expected result	1. 系統能順利統計景點出現次數。 2. 景點統計正確的輸出結果，把資料輸出到資料儲存模組做下一階段的運作。	
Cleanup		

4.1.2 IT2 測試案例

目的：

(1) 驗證確認讀取景點統計模組資料。

(2) 驗證能產生出不頻繁景點做待受保護資料。

表 4. IT2 測試案例

Identification	IT2	
Name	整合資料儲存模組與不頻繁景點模組	
Tested target		
Reference		
Severity		
Instructions	*Actor actions*	*System responses*
		1. 確認上一個測試案例輸出的景點統計到資料儲存做讀取動作。
		2. 確認檔案讀取後，將資料產生出不頻繁景點，並確保產生資料無誤。
		3. 將產生出來的不頻繁打卡資料稱做待受保護資料，替下一階段做資料處理。
Expected result	1. 系統能夠讀取到景點統計在資料儲存模組的資料。 2. 確認產生不頻繁景點。	
Cleanup		

4.1.3 IT3 測試案例

目的：

(1) 驗證能保護不頻繁景點資料。

(2) 確保有效保護不頻繁打卡資料外，同時也要考慮到本身資料原始的特性。

表 5.IT3 測試案例

Identification	IT3	
Name	整合不頻繁景點模組與保護不頻繁景點模組	
Tested target		
Reference		
Severity		
Instructions	*Actor actions*	*System responses*
		1.針對待受保護資料做泛化。
		2.結合頻繁景點與不頻繁景點找到共同特性，做泛化動作。
		3.最終，將泛化結果輸出到最原始打卡資料中，達到匿名保護的效果。
Expected result	1. 系統能針對較少出現次數的打卡資料做有效的保護。 2. 同時也要確保本身資料可用性。	
Cleanup		

4.2 接受度測試案例 (Acceptance Testing Cases)

4.2.1 AT1 測試案例

目的：

(1) 驗證是否能從打卡資料中探勘出不頻繁景點做待受保護的動作。

表 6.AT1 測試案例

Identification	AT1
Name	使用者初步使用本系統
Tested target	
Reference	UIR-001、UIR-002、IIR-001、IIR-002、IIR-003
Severity	
Instructions & Expected result	1. 針對所產生打卡的記錄做景點統計。 2. 系統能夠利用 FP-Growth 探勘出不頻繁景點。 3. 系統確認出不頻繁景點做待受保護動作。
Cleanup	

4.2.2 AT2 測試案例

目的：

(1) 驗證從待受保護資料做後續動作，利用泛化方式保護不頻繁景點資料。

表 7.AT1 測試案例

Identification	AT1
Name	使用者後續保護本系統
Tested target	
Reference	UIR-003
Severity	
Instructions & Expected result	1. 保護不頻繁景點，方法透過結合不頻繁打卡資料及頻繁打卡資料找到共同特性，達到泛化方式保護不頻繁景點。最終將打卡資料加入原始打卡記錄之中。 2. 確保本身資料可用性，取得保護資料與尚未保護資料的平衡點，最終保留本身資料特性。
Cleanup	

5. 測試結果與分析 (Test Results and Analysis)

5.1 整合測試案例 (Integration Testing Cases)

表 8.整合測試案例結果

Test Case #	Results (Pass/Fail)	Comment
IT1	Pass	
IT2	Pass	
IT3	Pass	

5.2 接受測試案例 (Acceptance Testing Cases)

表 9.接受度測試案例結果

Test Case #	Results (Pass/Fail)	Comment
AT1	Pass	
AT2	Pass	

Appendix A： 追朔表 Traceability

A.1. 子系統 vs. 測試案例 (Subsystems vs. Test Cases)

表 10.Subsystems vs. Test Cases Traceability Table

Test Cases / Subsystems	IT1	IT2	IT3
UIR-001	#		
UIR-002	#		
UIR-003			#
EIR-001			#
EIR-002			#
ACAPD-IIR-001	#		
ACAPD-IIR-002		#	
ACAPD-IIR-003		#	

Appendix B： 參考資料 (References)

[1] L. Sweeney, "k-anonymity: A model for protecting privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems,* vol. 10, no. 05, pp. 557-570, 2002.

[2] C. Dwork, "Differential privacy," in *Encyclopedia of Cryptography and Security*: Springer, 2011, pp. 338-340.

[3] E. Baralis, L. Cagliero, T. Cerquitelli, and P. Garza, "Generalized association rule mining with constraints," *Information Sciences,* vol. 194, pp. 68-84, 2012.

[4] X. Huang, "On global sequence alignment," *Computer applications in the biosciences: CABIOS,* vol. 10, no. 3, pp. 227-235, 1994.

[5] V. Garg, S. Arora, and C. Gupta, "Cloud computing approaches to accelerate drug discovery value chain," *Combinatorial chemistry & high throughput screening,* vol. 14, no. 10, pp. 861-871, 2011.

[6] X. Lin, "Mr-apriori: Association rules algorithm based on mapreduce," in *Software Engineering and Service Science (ICSESS), 2014 5th IEEE International Conference on*, 2014, pp. 141-144: IEEE.

[7] L. Chunqing, "Apriori Algorithm Optimization Study Based on MapReduce," 2015.

[8] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in *ACM Sigmod Record*, 2000, vol. 29, no. 2, pp. 1-12: ACM.