擴充關聯資料模式支援沙盤推演式資料分析

許中川
雲林科技大學資訊管理學系

摘 要

資料庫系統已經是普遍使用的資料儲存工具。企業資料庫中儲存著大量資料，其中隱藏著許多有用的資訊，有效的分析及妥善的運用，可協助企業制訂具有競爭力的策略。管理資訊系統的操作性資料庫無法有效的支援決策分析。近幾年，資料倉儲技術將不同來源的操作性資料庫整合成資料倉庫，以便利支援決策分析。一些研究提出用包含維度綱要及事實綱要的多維度資料模式塑模資料倉庫，然而固定式的維度綱要，不易調整，缺乏彈性及親和性，無法對沙盤推演式資料分析提供良好的支援。本論文探討強化資料庫管理系統對線上資料分析的能力。我們提出以無綱要式的屬性階層擴充關聯資料模式，並且擴充結構化查詢語言，以便利用儲存在屬性階層的資料，進行高層次的資料查詢與分析。另外，我們提出趨勢聚集函數，便利計算區段時間內，數值屬性的資料值變化趨勢。最後，我們開發一視覺化線上資料查詢及分析雛形系統，驗證方法的可行性以及對沙盤推演式資料分析的操作親和性。

關鍵詞：線上資料分析處理、擴充式關聯資料模式、沙盤推演式分析、決策支援

Extending Relational Data Model to Support What-If Data Analysis

Chung-Chian Hsu
Department of Information Management
National Yunlin University of Science and Technology

ABSTRACT

Database systems have become a popular tool for data storage. Enterprise's databases store massive data in that useful information may hide. Properly analyzing those data may help an enterprise in competitive strategies making. Operational databases of management information systems can not effectively support decision making. In recent years, the emerging technology, data warehousing, provides a means to integrate various operational databases into a data warehouse to facilitate on-line data analysis. Some research proposed using multidimensional data model that includes dimension schemata and fact schemata for data warehouse. However, static dimension schemata lack of flexibility and friendliness can not well support what-if data analysis. This paper discusses strengthening the capability of on-line data analysis of database management systems. We propose extending relational data model with schemaless attribute hierarchy and also extending conventional structured query language in order to utilize the information stored in attribute hierarchies for high-level query and analysis. Besides, we propose a trend aggregation function for computing the trend of a numerical attribute. Finally, we develop a visual on-line query and analysis prototype to prove the feasibility of the proposed model and to show better support to what-if analysis.

Keywords: On-line Analytical Processing, Extended Relational Data Model, What-If Analysis, Decision Support

壹、緒論

一、動機與目的

在現今的企業，資料庫已經是普遍使用的資料儲存工具。企業資料庫中儲存著大量資料，例如在銷售資料庫中各個通路的每日銷售資料，以及備份資料庫中的大量歷史銷售資料。這些日積月累巨量資料已成爲企業的一項重要資產，其中隱藏著許多有用的資訊，有效的分析及妥善的利用，可協助企業制訂具有競爭力的策略。

交易處理系統的操作性資料庫無法有效的支援決策分析 (Codd 1993;Finkelstein 1995;Kimball 1995)。傳統的操作性資料庫記錄每日交易資料，著重資料一致性、節省儲存空間、方便修改，以避免更正異常 (update anomalies) (Elmasri 1994)，因此操作性資料庫的設計都經過正規劃處理，將資料表格切割，以達到較高階的正規型態。一個典型的帳單處理系統的資料庫至少 50 個，也可能多達 200 個資料表格。雖然正規化後的資料庫，方便資料輸入及更新，但是對正規化的資料庫進行資料分析是非常沒有效率的，對決策活動的支援效果不彰。

近年來，有不少研究透過資料倉儲 (data warehousing) (Chaudhuri 1997; Widom 1995;Inmon 1996) 及線上分析處理 (on-line analytical processing, OLAP) (Kimball 1995;Li 1996;Gray 1997;Gupta 1995)，探討強化資料庫管理系統對決策活動的支援。資料倉儲整合企業內日常交易處理系統的操作性資料以及其他不同來源的資料，例如網際網路上收集到的資料或購至市場調查公司的競爭對手資料，形成主題導向、整合性的資料倉庫 (data warehouse)，然後進行各種線上分析處理，包括向上合成 (roll-up)、向下展開 (drill-down)、切割、排名及比較等 (Kimball 1996)。有一些研究探討如何塑模多維度的資料倉庫。大部分研究提出以維度綱要塑模維度資料，以事實綱要塑模事實資料 (Chaudhuri 1997;Cabibbo 1997; Gyssens 1996)。然而，以固定式的維度綱要表示維度資料類別架構，不容易彈性調整分類，無法對沙盤推演式 (what-if) 的決策活動，提供良好的支援。另外，對複雜的維度資料維度綱要無法妥善的塑模。

本研究擬提出一適合沙盤推演式資料分析的邏輯資料模式。讓分析人員容易表示複雜情況下的維度資料，同時讓分析人員很容易調整維度資料，以便從不同角度觀察資料，進行沙盤推演式的資料分析。我們提出以屬性階層擴充關聯式資料模式，並且擴充結構化查詢語言，以便能利用儲存在屬性階層內的維度資料，進行高層次的查詢。另外，在資料分析方面，提出一個新的聚集函數，配合擴充的資料模式，進行數值資料的趨勢分析以及異常分析。我們以提出的理論，建構一視覺化資料查詢與分析雛形系統，驗證擴充式關聯資料模式的分析彈性及操作親和性。

二、問題陳述

爲便利從不同角度分析存放在資料倉庫的資料，許多研究探討如何塑模多維度的資料倉庫。大部分的多維度資料塑模研究，將多維度資料庫的資料分成兩大類：維度資料及事實資料，提出利用維度綱要及事實綱要塑模維度資料及事實資料。然而，以固定式的維度綱要表示維度資料的類別架構，不具彈性，要調整維度架構時，必須重新設計維度綱要及輸入資料。因此，無法對沙盤推演式的決策活動，提供最好的支援。此外，對複雜的維度資料，維度綱要無法妥善的塑模。

對於複雜的維度資料，定義固定式的維度綱要會遇到兩個困難：首先，當維度

資料的階層深度不一時，以固定欄位的維度綱要無法適當的表示。以目前大部分商用資料倉儲系統所用的星狀綱要（ star schema ）爲例。假設有一個維度綱要 Location 及事實綱要 Sales 如下：

Location(Store, City, Area)
Sales(Product, Date, Time, Store,
Unit, Price, Amount)

Location 綱要將銷售地點的類別架構，分成店舖 Store 、所在的縣市 City 及北中南大三區域 Area 。然而，縣市大小不一，嘉義市僅幾十萬人，台北市卻有幾百萬人，有必要再細分行政區。這種維度資料的分類階層深度不一的情形，造成定義維度綱要的困難。若將 Location 綱要擴充加入行政區，會造成部分資料記錄的行政區屬性，儲存空值（如嘉義市）。在屬性值衆多且特性差異懸殊的屬性，類似的問題更加嚴重。例如 Product 屬性，針對大型百貨公司或量販店而言，產品種類繁多，性質差異很大，很難以統一架構的產品維度綱要，表示產品維度的階層架構。

另外一個問題是，類別性質差異大的分支，很難以需要統一分類準則的維度綱要表示。以大型量販店賣場產品爲例，在產品維度資料的分類深度一，可將產品依「型態」分成食、衣、住、行四大類別。於分類樹的深度二時，由於產品特性差異懸殊，各有不同的分類角度，很難再定義一個有意義的共同分類名稱，而且深度越大，困難度越高。有公司採用星狀綱要，將產品維度綱要定義成 Product(產品編號，小分類，中分類，大分類）。這樣的屬性名稱在語意上並不是很妥善，然而對於複雜的產品資料，實在很難定義一個語意清楚的產品維度綱要。

觀察某段期間數值資料的變化情形，爲一種常見的資料分析。本文中，我們稱爲趨勢分析。例如分析過去十年來，台灣每年的經濟成長率；或以銷售資料而言，如下列的查詢一，分析人員想觀察八十六年下半年度，所有碳酸飲料總和銷售的逐月銷售量增減百分比。

查詢一：八十六年度七月到十二月間，碳酸類飲料總和的月銷售趨勢。

針對查詢一，使用目前的結構化查詢語言，無法直接處理，而需配合第三代或第四代語言。主要的問題：(1) Date 屬性值包括年月日，無法使用群聚（ grouping ）及加總聚集函數（ SUM ），直接加總碳酸飲料的月銷售總額。(2)沒有聚集函數（ aggregation function ）直接計算逐月的銷售量增減情形。

配合門檻值，趨勢分析可進一步擴充成異常分析，例如查詢二找出八十六年度碳酸飲料中，月銷售變化量超過-5% 的飲料。

查詢二：八十六年度，月銷售量減少超過 5% 的碳酸飲料。

趨勢分析及異常分析在概念上都是很直覺，同時也是極爲普遍的分析。然而直接使用結構化查詢語言，卻是非常複雜，或甚至不可行，需配合第三代或第四代語言，撰寫訂製程式。訂製程式通常缺乏彈性，且不易維護，對沙盤推演式決策活動無法提供良好的支援。

貳、以屬性階層擴充資料模式

對資料分析人員而言，以維度綱要塑模資料分類架構，不易調整，缺乏彈性。

我們提出的方法是無綱要式：直接以屬性階層擴充關聯資料模式。擴充式的關聯資料模式，其超資料（ meta data ）主要包括四個組成 <D, AH, RS, IC> ：

1.D ：領域定義 (domain definitions) 集合。
2.AH：屬性階層 (attribute hierarchies) 集合。
3.RS ：關聯綱要 (relation sche-mata) 集合。
4.IC ：完整性限制 (integrity cons-traints) 集合。

領域定義集合定義各個領域的合法值。資料表格的每一屬性關聯某一領域，而資料庫中每一個屬性的屬性資料值必須是該所屬領域合法值的元素。關聯綱要集合包含各個關聯資料表格的定義：包括資料表格名稱、屬性名稱、屬性領域等。屬性領域爲上述領域定義集合之一元素。完整性限制集合包含結構完整性限制及語意完整性限制，定義一個合法關聯資料庫的規則。一個合法的資料庫，各個資料表格的每一筆資料記錄，不得違反任何一條完整性限制。

資料表格的每個屬性除了關聯到某一領域，也同時關聯到一個相同名稱的屬性階層。兩個不同的表格中，若有相同的屬性名稱，則屬性階層名稱上再加上資料表格名稱以便區別。屬性階層主要定義屬性值的較高層次的一般化概念。屬性階層可視爲一棵階層樹，葉節點儲存屬性值。一個節點的父節點，儲存更一般化的概念或稱爲類別資訊，例如「可口可樂」是一種「碳酸」飲料。根節點的資料值爲「任何」，是最大的類別。屬性階層的節點深度是指節點到根節點的距離。根節點的深度爲零，根節點的子節點的節點深度爲一。依此類推，每往下一層，深度增加一。

屬性階層可以用來塑模資料倉庫中的維度資料。一個產品維度的類別資料，以產品屬性階層表示（如圖 1 ），在「產品」領域內，首先，分成「食」、「衣」、「住」及「行」四類。然後，再繼續細分。「飲料」包含有「碳酸」飲料、「茶類」飲料及「礦泉水」飲料等。圖 2 爲一個日期屬性階層樹。

以無綱要式屬性階層塑模維度資料的特色是，階層的分支深度可以不一樣，以及沒有命名綱要屬性名稱的問題。這兩點特色使得屬性階層較容易塑模複雜情形下的維度資料。另外，分析人員可以直接調整屬性階層，從不同角度分析資料，對沙盤推演式的決策活動提供較好的支援。

一個擴充後關聯資料庫包括兩部分：超資料（ meta-data ）及資料表格集合。資料表格集合包括原始資料表格及一般化資料表格（ generalized relation ），如表 1 及表 2 。一般化資料表格是由原始資料表格或其它一般化資料表格，經過一般化處理，所得的資料表格。一般化處理是針對資料表格的一個或一個以上的屬性，將屬性值以其屬性階層中，較高層次的類別資訊取代。

參、擴充結構化查詢語言

配合關聯資料模式的擴充，本節進一步擴充結構化查詢語言，以充分利用儲存在屬性階層的資訊。我們增加 GENERALIZE 指令及 PARENT 函數，以便能夠進行高層次資料查詢及統計。此外，提出 TREND 函數，以便進行趨勢分析及異常分析。

一、一般化指令

一般化指令根據屬性階層，將指定的

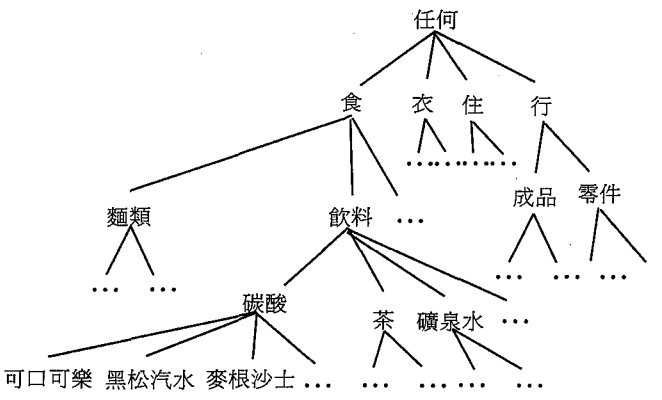

圖 1 ：產品屬性階層

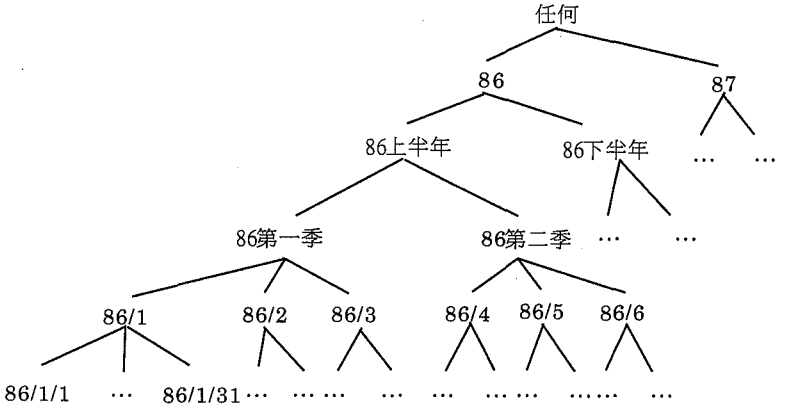

圖 2 ：日期屬性階層

表1：原始銷售資料表格

Product	Date	Time	Store	Unit	Price	Amount
可口可樂	86/02/13	17:34	S510	2	20	40
大福麵條	86/02/13	17:34	S510	3	30	90
黑松汽水	86/02/13	17:40	S510	1	20	20
…	…	…	…	…	…	…

表2：Product, Date及Store一般化後的一般化銷售資料表格

Product	Date	Time	Store	Unit	Price	Amount
飲料	86/02	17:34	雲林	2	20	40
麵類	86/02	17:34	雲林	3	30	90
飲料	86/02	17:40	雲林	1	20	20
…	…	…	…	…	…	…

資料表格屬性的屬性值一般化到指定的深度。指令語法如下：

GENERALIZE <attribute list> TO <depth list> FROM <table name>
其中
<attribute list> ：屬性串列，決定要對哪一些屬性之資料值進行一般化
<depth list> ：深度串列，用來指定某個屬性將要被一般化到何種程度
<table name> ：欲一般化的資料表格

屬性串列及深度串列中的成員個數相等，且具有兩兩對應的關係。例如，屬性串列 <Product ， Date> 及深度串列 <1,4> ，表示要將 Product 屬性根據對應的 Product 屬性階層，一般化到深度爲一的程度；以圖 1 及圖 2 的屬性階層爲例，也就是要把各類產品一般化到「食」「衣」「住」或「行」的層次，而要將 Date 屬性根據 Date 屬性階層，一般化到深度爲四，亦即以「月」爲最小的時間單位。一般化資料表格的屬性名稱可以沿用原屬性名稱或利用 AS 關鍵字定義新屬性名稱。下列指令將銷售資料表格中的產品及銷售日期欄位，分別一般化到深度一及深度四：

GENERALIZE Product, Date TO 1, 4
FROM Sales

GENERALIZE 命令除了可以單獨使用外，亦可配合 SELECT ，形成複合指令。例如：

SELECT Product, Date, SUM(Amount)
FROM Sales WITH Product, Date
GENERALIZED TO 1, 4
WHERE Product = " 食 " AND Date = {86/12}
GROUP BY Product, Date

重新命名屬性，可增加指令的可讀性：

SELECT Category, Month,
SUM(Amount)
FROM Sales WITH Product, Date
GENERALIZED TO 1 AS Category, 4 AS Month
WHERE Category = " 食 " AND Month = {86/12}
GROUP BY Category, Month

二、父節點函數

資料表格屬性值的高層次類別資訊儲存於屬性階層中， PARENT 函數提供向上航行屬性階層樹的功能。 PARENT 函數主要的作用在於，擷取屬性階層某節點的父節點資料。基本語法爲：

PARENT(<attribute>)

語意爲針對某一資料記錄的 <attribute > 屬性，根據該屬性目前的屬性值，於其對應的屬性階層樹中，找出其父節點資料值。 PARENT 函數可以串接，也就是可以 PARENT(PARENT(<attribute>)) 之型式表示與某屬性值深度相差二的高層次資料值。

範例一：針對資料記錄 < 可口可樂， 86/02/13 ， 17:34 ， 2 ， 20 ， 40> 及圖 1 的產品屬性階層， PARENT(Product) 函數值爲「碳酸」。 PARENT(PARENT (Product)) 函數值爲「飲料」。

PARENT 函數配合屬性階層，可以容易地實施線上分析處理中，常用的切割處理 (Slice-and-Dice) (Chaudhuri1997)，也就是篩選某特定主題的資料。下列指令

篩選所有關於飲料的交易資料。

```
SELECT *
FROM Sales
WHERE PARENT(PARENT(Product))
= "飲料"
```

三、趨勢聚集函數

傳統結構化查詢語言提供資料分析的功能，包括加總 (SUM) 、平均 (AVG) 、計數 (COUNT) 、最大值 (MAX) 及最小值 (MIN)(IS1992) 。這些功能僅能夠做簡單的資料分析，對於較複雜的分析，無法以簡潔的指令表示。針對數值資料的變化分析，我們提出 TREND 聚集函數。

TREND 聚集函數是用來計算某時間區段內，數值資料的變化趨勢；也就是在指定的時間區段內，數值資料的增減百分比。使用到的參數包括要觀察的數值欄位及要觀察的時間區段。例如八十六年下半年，所有碳酸飲料總和銷售的趨勢；也就是八十六年七月到十二月碳酸飲料銷售量的逐月增減情形。 TREND 函數的使用語法表示如下，爲降低語法複雜度，此處僅列出和本研究有關的部分，其餘部分和標準的結構化查詢語言語法相同：

```
SELECT <attribute list> TREND
        (<numeric attribute>)
FROM <table name> [WITH
      <attribute list> GENERALIZED
     TO <depth list>]
WHERE <time attribute> FROM<t1> TO
      <t2> [{AND (<parent condition
      > | <other condition>)}]
[GROUP BY <grouping attributes>]
[HAVING <group selection condition> ]
  <parent condition> ::=<parent function>
  <comparison operator>
  <attribute value>
  <parent function> ::=
      PAENT(<attribute>) |
      PARENT(<parent function>)
```

<numeric attribute> 爲要計算趨勢的屬性，必須是數值型態。 <t1> 及 <t2> 分別爲要觀察的時間區間的起始點及終止點。給予一表格 T 及數值屬性 A ，時間區段起始值 tstart 及終止點 tend ，表格的資料按時間屬性由小至大排列。 TREND 函數的運算結果爲一串列 $<x_{tstart}, x_{tstart+1}, ..., x_{tend}>$ ，串列中的每一個元素，表示兩個連續時間單位， A 屬性值的增減百分比。亦即

$$x_i = \begin{cases} 0\% & if\ i = t_{start} \\ (v_i - v_{i-1})/v_{i-1} * 100\% & if\ t_{start} < i \le t_{end},\ and \\ & v_{i-1} \notin \{0, UNLL, missing\} \\ NULL & if\ v_{i-1} \in \{0, NULL, missing\} \end{cases}$$

其中

v_i ：i 時間點的 T.A 屬性值

時間區段起始點的函數值設爲零。當 v_i 的值爲零、空值 (NULL) 或沒有值時，則定義函數值爲 NULL 。

範例二：緒論中所提的查詢一，分析八十六年度七月到十二月間，所有碳酸飲料的總合銷售趨勢。查詢指令及一個虛擬的查詢結果如下：

```
SELECT Category, Month,
       TREND(Amount) AS
       SalesTrend
FROM Sales WITH Product, Date
     GENERALIZED TO 3 AS
      Category, 4 AS Month
WHERE Month FROM {86/7} TO {86/
```

12} AND Category = " 碳酸 "
GROUP BY Category, Month

虛擬的結果：

Category	Month	SalesTrend
碳酸	86/7	0 %
碳酸	86/8	2 %
碳酸	86/9	6 %
碳酸	86/10	-2 %
碳酸	86/11	-3 %
碳酸	86/12	-1 %

本查詢無法直接使用傳統結構化語言直接表示，必須用第三代或第四代語言寫成訂製程式，而無法直接進行線上分析。使用擴充後的資料模式及擴充後的查詢語言，可以單一查詢指令直接表示。緒論中提到的查詢二異常分析，可用下列的擴充式查詢指令表示。另外，查詢三分析八十四至八十六的各年度中，各類飲料（碳酸、茶類、礦泉水等）的銷售量。

查詢二：八十六年度，月銷售量減少超過 5% 的碳酸飲料。

```
SELECT Product, Month,
       TREND(Amount)
FROM Sales WITH Date
     GENERALIZED TO 4 AS
     Month
WHERE Date FROM {86/1} TO
      {86/12} AND
      PARENT(Product) = " 碳酸 "
GROUP BY Product, Month
HAVING TREND(Amount) <-5%
```

查詢三：八十四至八十六的各年度中，各類飲料的銷售量？

```
SELECT DrinkCategory, Year,
       SUM(Amount)
FROM Sales WITH Product, Date
     GENERALIZED TO 3 AS
     DrinkCategory, 2 AS Year
WHERE Parent(DrinkCategory) =
      " 飲料 " AND Year ≥ 84
      AND Year ≤ 86
GROUP BY DrinkCategory, Year
```

肆、查詢指令處理

一、GENERALIZE演算法

資料表格的一般化處理，我們提出三種方式。第一種方式是將屬性階層存放在一般檔案中。一般化處理時，讀入主記憶體，利用節點及鏈結安排成樹狀資料結構。屬性階層樹的每一節點主要有四個組成：節點資料值（Value）、子節點串列指標、兄弟節點指標、父節點指標(Parent)、深度值(Depth)。節點資料值儲存原始資料表格的資料值，如「可口可樂」；或高層次的類別資訊，如「碳酸」飲料。子節點串列指標及兄弟節點指標是爲了由上往下搜尋某指定的節點資料值。每一個節點的子節點串列指標指向第一個子節點（最左邊的子節點），兄弟節點指標向右指向同一層節點。每一節點都有一指標指向父節點，且都有一個深度值，表示該節點於屬性階層樹的深度。父節點指標主要用來擷取父節點資料值或者是由下往上爬升到指定的屬性階層深度，以便進行一般化處理。

一般化處理是針對資料表格中的每一筆資料記錄，依據欲一般化的屬性串列中的每個屬性逐一進行屬性值一般化。其步驟大致爲：在屬性階層樹中利用深度搜尋法，從根節點由上往下搜尋，找出資料記錄的屬性值在屬性階層樹中對應的節點。若該節點的節點深度大於欲一般化深度，則以父節點指標往上爬升，直到指定的一

般化深度，然後以該節點資料值取代原屬性值。演算法如下：

輸入：資料表格、欲一般化之屬性 A、欲一般化之深度 Depth 、屬性 A 的屬性階層
輸出：一般化後的資料表格。
演算法一：
將資料表格根據 A 屬性排序
FOR t ∈ 資料表格
IF t.A 和前一筆資料 t'未一般化前有相同的 A 屬性值
THEN t.A ← t'.A
ELSE
深度搜尋 A 屬性的屬性階層樹，找出節點 Node ，且
Node.Value=t.A
WHILE (Node.Depth>Depth)
Node ← Node.Parent
t.A ← Node.Value

本演算法有三個主要的處理：排序、逐筆一般化資料記錄、搜尋屬性階層樹。主要的時間瓶頸在於，產品屬性階層樹龐大，搜尋時間頗長。根據實驗顯示，我們以學校員生消費合作社的 7795 筆交易資料，及依據產品建立一棵 297 節點的產品屬性階層樹，在個人電腦 Pentium200, 128MRAM 的環境，進行產品屬性一般化，結果大約需 137 分鐘。

第二種方法，我們嘗試將屬性階層轉以屬性階層資料表格儲存，直接在屬性階層表中搜尋指定的屬性值以及指定的階層深度。屬性階層表包含三個欄位：節點資料、父節點資料及節點深度（如表 3 ）。屬性階層表根據節點資料加以排序，以便加快搜尋速度。除非屬性階層樹修改，否則轉換處理只需做一次，爾後可以直接使用屬性階層表。一般化處理時，將資料表格依欲一般化的屬性，加以排序，以便利同時一般化具有相同屬性值的資料紀錄。然後，在屬性階層表中進行搜尋，找出資料表的資料記錄屬性值在屬性階層表中的對應節點資料，接著以其父節點資料值取代原屬性值。重複此步驟，直到指定的一般化深度。其演算法如下：

輸入：資料表格、欲一般化之 A 屬性、欲一般化之深度 Depth 、A 屬性的屬性階層表
輸出：一般化後的資料表。
演算法二：
將資料表根據 A 屬性進行排序
FOR t ∈ 資料表格

表3：依據圖1，用於第二種一般化方法的產品屬性階層表

Node	Parent	NodeDepth
…	…	…
可口可樂	碳酸	4
任何	NULL	0
成品	行	2
衣	任何	1
住	任何	1
食	任何	1
茶	飲料	3
…	…	…

表4：依據圖1，欲將產品屬性階層一般化到深度二時，轉換後的屬性階層表

Node	GNode
…	麵類
碳酸	飲料
可口可樂	飲料
黑松汽水	飲料
麥根沙士	飲料
…	…
…	成品
…	零件
…	…

```
IF t 與前一筆資料 t' 未一般化前
有相同的 A 屬性值
THEN t.A ← t'.A
ELSE
    temp ← t.A
    搜尋屬性階層表，找出資料記
    錄 t" 且 t".Node = temp
    WHILE
    ((t".NodeDepth-1)>Depth)
        temp ← t".Parent
        搜尋屬性階層表，找出資料
  記錄 t" 且 t".Node = temp
 t.A ← temp
```

根據實驗顯示，上述的 7795 筆資料進行同樣的一般化處理，只花了 32 秒時間。主要改善的原因是，屬性階層轉以表格方式儲存，並且經過排序，可以加快搜尋速度。我們進一步加大資料量，改用 96490 筆資料進行同樣的一般化處理，結果花了 114 秒時間。

第三種方法，根據查詢指令中，欲一般化的深度，動態地將屬性階層樹中，該深度及以下的節點轉成屬性階層表，屬性階層表只包括兩個欄位，一爲節點 Node ，存放一般化深度以下的節點資料。另一爲一般化後節點 GNode ，存放指定深度的節點資料（如表 4 ）。然後直接利用結構化查詢語言的合併運算（ join ），以屬性階層表的節點資料和資料表的被一般化屬性做等位合併運算（ equal-join ），並以屬性階層表中的一般化後節點資料屬性取代資料表中的被一般化的屬性。例如，要將銷售資料表的產品屬性一般化到圖 1 的深度二，則動態的將深度二以下的資料轉成表 4 。然後將表 4 的 Node 屬性和銷售資料表的產品屬性，進行等位合併運算。並投射（ project ） GNode 屬性取代原資料表的產品屬性。

第三種方法需要用到等位合併運算，我們將資料移到 DEC AlphaServer 4000 的 Oracle 資料庫系統，以利用 Oracle 系統進行合併運算，前端的處理還是使用個人電腦。在此次實驗中，我們進行較多的測試，希望進一步觀察資料量和一般化處理速度之間的關係。資料料量由一萬筆開始，每次增加一萬筆。結果顯示資料筆數和處理速度呈線性關係，如圖 3 。對 96490 筆資料進行一般化處理花了 21.24 秒。

二、PARENT函數演算法

PARENT 函數之功能在於傳回某個屬性資料值的父節點資料值。 PARENT 函數指令的處理與一般化演算法類似；給予一筆資料記錄，一個指定的屬性和其屬性值以及該屬性的屬性階層。如果屬性階層是以屬性階層樹的形式讀入存放在主記憶體，則基本處理過程是，從根節點開始，利用深度搜尋法搜尋屬性階層樹，尋找節點資料值和該屬性資料值相同的節點，然後利用父節點指標，擷取並傳回父節點資料值。如果屬性階層是如上述第二種方法，轉換成節點、父節點及節點深度的三欄位屬性階層表，則基本處理過程是：利用搜尋屬性階層表的節點欄位，尋找欄位資料值和該屬性資料值相同的節點，然後傳回父節點欄位的欄位值。

三、趨勢查詢處理

聚集指令分成純量聚集（ scalar aggregates ）及聚集函數 (aggregate functions) (Graefe 1993) 。純量聚集的結果爲一純量數值，例如「所有員工的薪資總和」。聚集函數運算結果爲一關聯資料表格，例如「各個部門的部門員工薪資總和」。 TREND 函數爲聚集函數。聚集指令的計算處理，基本上有三種方式：巢狀迴圈法、排序法及雜湊法 (Graefe 1993; Elmasri 1994) ，排序法是目前商用資料庫管

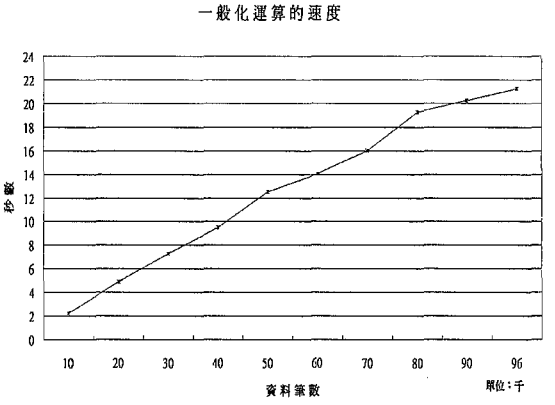

圖3：資料筆數與一般化處理效率的關係

理系統最統最常用的方法。本節探討利用排序法處理 TREND 聚集查詢指令。有一些論文針對聚集指令最佳化，做更進一步的探討，提出特殊的運算技巧，本文目前不作討論 (Klug 1982;Muralikrishna 1992; Yan 1994-95;Chaudhuri 1994-96)。

針對一條趨勢查詢指令，處理程序爲：(1)如果查詢指令中，有內嵌的一般化指令，則先處理 GENERALIZE 部分。亦即對分析人員所指定的屬性資料進行一般化處理，將屬性資料值一般化到所指定的深度。(2)將資料表格依據時間屬性進行由小到大的排序。然後，篩選符合查詢區間的資料記錄。(3)進一步篩選符合 WHERE 子句其它限制條件的資料記錄。(4)進行群聚（ grouping ）：根據群聚串列及趨勢屬性，投射屬性。接著將資料記錄依群聚屬性排序，使得同群組資料記錄相鄰排列，以利聚集運算。(5)加總同一群組的趨勢屬性值，並進行去除重複處理，也就是根據群聚屬性合併同組資料記錄，然後加上加總後的趨勢屬性值，成爲一筆群聚後的資料記錄。(6)計算連續時間區段的趨勢屬性資料值變化百分比，結果存入群聚後資料記錄的趨勢屬性。(7)如果有 HAVING 條件子句，進行趨勢的異常分析，則進一步依據 HAVING 條件，將符合條件的群聚資料記錄篩選出來。接著依據 SELECT 的屬性串列，投射選擇的屬性。趨勢查詢演算法整理如下：

輸入：資料表格、屬性階層、趨勢查詢指令（包括趨勢屬性、一般化屬性串列、一般化深度串列、起始時間、終止時間、群聚串列及其他篩選條件）。

輸出：時間區間之銷售趨勢資料表格。

演算法：

1. 若有屬性值須一般化，則呼叫 GENERALIZE 程序進行一般化。
2. 依據時間屬性，進行由小到大的資料記錄排序。然後，篩選符合查詢區間的資料記錄。
3. 篩選符合 WHERE 子句中其它條件的資料記錄。
4. 根據 GROUP BY 的屬性串列及趨勢屬性，投射屬性及排序。
5. 加總同群組的趨勢屬性的資料值，並合併同群組的資料記錄。
6. 計算連續時間區段的趨勢屬性資料變化比率，儲存結果於趨勢屬性。
7. 篩選聚集後符合條件的資料記錄，然後投射選擇的欄位。（若進行異常分析）

以「八十六年度，月銷售量減少超過

5% 的碳酸飲料」查詢爲例，說明查詢執行程序：

```
SELECT Product, Month
       TREND(Amount)
FROM Sales WITH Date
       ENERALIZED TO 4 AS
       Month
WHERE Month FROM {86/1} TO
       {86/12} AND
       PARENT(Product) = "碳酸"
GROUP BY Product, Month
HAVING TREND(Amount) < -5%
```

處理程序：(1)進行屬性值一般化，將銷售日期由以日爲單位一般化到以月爲單位。(2)以時間屬性對資料表格排序，越新近資料記錄，排在越後面。篩選月份介於{86/1} 及 {86/12} 之間的資料記錄。(3)篩選出 PARENT(Product) 爲「碳酸」的資料記錄。(4)投射 Product 、 Month 及 Amount 屬性，並根據 Product 及 Month 屬性排序資料記錄。(5)進行群聚加總，也就是 Product 及 Month 資料值相同的資料記錄，加總 Amount 屬性值。然後，同一群組資料紀錄，只保留一筆群聚屬性值及加總的趨勢屬性值。(6)計算 TREND (Amount) 。(7)將 TREND (Amount) 小於-5% 的資料記錄，其 Product 及 Month 屬性及 TREND(Amount) 資料值篩選出來。

伍、視覺化雛形系統

爲驗證提出的擴充式關聯資料模式的可行性及支援沙盤推演式分析的便利性。我們開發一視覺化資料查詢與分析系統（ Visual Data Query and Analysis System, or VDQAS ）。VDQAS 採用主從架構，目前後端的資料庫系統爲 Oracle ，平台爲 DEC AlphaServer 4000 ，負責資料存取及一般化運算。前端是以 Borland C++ Builder 開發的視覺化操作介面，平台爲 Windows 個人電腦，負責視覺化輸入輸出及趨勢分析。雛型系統的功能包括視覺化屬性階層編輯器，擴充式結構化查詢語言編輯器，及視覺化輸出。本節說明各子系統功能，且以範例說明查詢過程。

一、屬性階層編輯器

VDQAS 的視覺化屬性階層編輯器，提供使用者建立一棵新的屬性階層樹，或從檔案中載入一棵已建立之屬性階層，並進行維護。圖 4 爲一棵建立中的產品屬性階層，使用者可以展開任意一個內部節

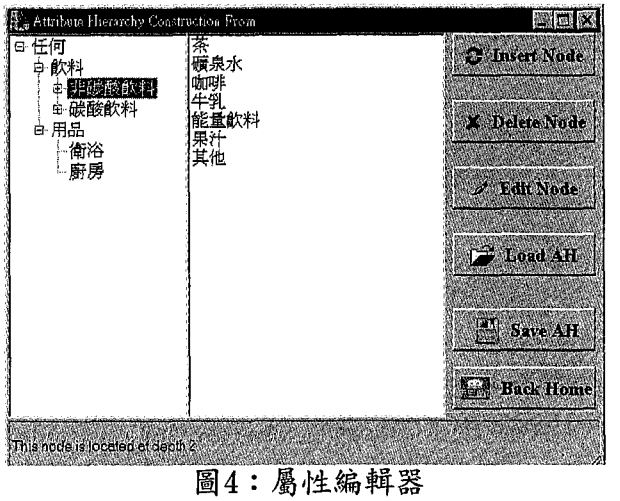

圖4：屬性編輯器

點，此節點下的子節點內容會出現在視窗的右邊。使用者可以新增、刪除節點，亦可以修改節點之內容；在狀態列上會顯示目前所選定之節點在屬性階層中的深度。使用者在修改屬性階層的節點資料後，可以將最後修改結果儲存至檔案。

屬性階層可視爲領域知識，可由專業分析人員動態建立及動態修改。也就是在進行資料分析時，可以線上調整屬性階層，從不同的角度，驗證各種假設，達到支援沙盤推演式資料分析的目的。爲提升使用的親和性，避免分析過程經常調整同一個屬性的屬性階層，VDQAS 允許分析人員儲存一個屬性常用的不同屬性階層。預設和該屬性名稱相同的屬性階層。然而，也可以透過介面，選擇使用其他的屬性階層，作不同的分類分析。

二、擴充式結構化查詢編輯器

VDQAS 的查詢編輯器（圖 5 ）可以編輯傳統的結構化查詢語言。此外，還有擴充的功能，包括資料表格一般化、內嵌一般化的查詢指令、PARENT 函數、TREND 趨勢分析函數。

資料表的一般化處理，可以透過編輯器編輯指令獨立執行。例如將產品屬性及日期屬性，根據對應的屬性階層分別一般化到第三深度及第四深度。一般化後的資料表，可以接著再編輯傳統的查詢指令，做進一步的查詢及分析。除此之外，也可以將一般化處理內嵌在傳統查詢指令內，成爲複合指令，由系統直接連續處理一般化以及查詢與分析。

三、視覺化查詢與分析

我們以範例說明一個內嵌一般化的趨勢查詢，並藉此展示視覺化的系統操作流程。由於銷售資料表中儲存的銷售產品資料，通常是條碼或是產品品名，而非類別資訊；此外，銷售日期包括年月日三個細項，所以本範例無法使用傳統結構化語言直接表示，必須用第三代或第四代語言寫成訂製程式，因而無法直接進行線上分析。使用擴充後的資料模式及結構化查詢語言，可以單一查詢指令直接表示。我們使用的測試資料爲學校員生消費合作社八十六年四月份到九月份的產品銷售資料，總共有 96490 筆。

範例三：八十六年四月至九月各類非碳酸飲料的銷售趨勢。

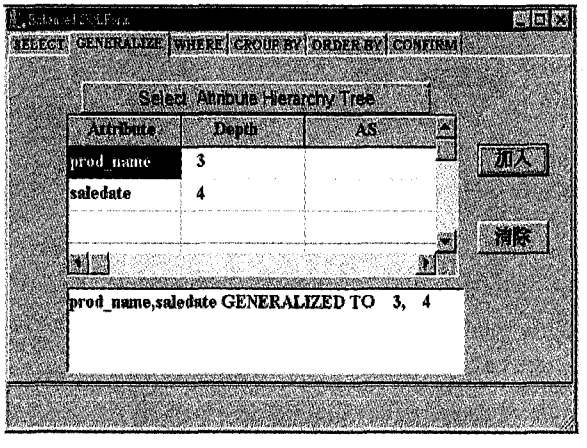

圖5：指定欲一般化的屬性及深度

```
SELECT Product, SaleDate
       TREND(Amount)
FROM Sales WITH Product ,
     SaleDate GENERALIZED
     TO 3 , 4
WHERE PARENT(Product) = " 非碳
     酸飲料 " AND SaleDate FROM
     {86/04} TO {86/09}
GROUP BY Product , SaleDate
```

步驟一、開啓所欲進行分析之資料庫及銷售資料表。

步驟二、在 SELECT 頁面，選擇屬性 Product, SaleDate 及聚集函數 TREND(Amount)。

步驟三、在 GENERALIZE 頁面，選擇欲進行一般化之資料欄位，並指定相對應之屬性階層及所要一般化的深度，如圖 5 。

步驟四、在 WHERE 頁面，指定篩選資料記錄的條件，如圖 6 。

步驟五、在 GROUP BY 頁面，指定欲進行群聚之資料欄位 Product 及 SaleDate 。

步驟六、CONFIRM 頁面可以確認編輯後的查詢指令，必要時可以再做修正。若沒問題，則執行查詢。

步驟七、對於結果的輸出，分析人員可以指定以資料表格方式呈現，或以圖形方式呈現，如圖 6 。

根據圖 6 顯示，查詢結果似乎有些異常的情況，在六月份至八月份之間的銷售量銳減，而九月份的銷售量則突然大增。觀察表格形式的趨勢資料及分析所使用的交易資料發現，六月份的確呈現負成長，因爲六月下旬學校期末考，然後接著放暑假。七月份與八月份僅有少數銷售資料，因爲七月份與八月份正值暑假期間，員生消費合作社並沒有每天營業。九月份學校開學後，交易大增，因此造成銷售趨勢突然大幅度成長。根據結果顯示，其中依次以牛乳、茶、果汁類增加的銷售幅度較多。

陸、相關研究

近年來，資料庫領域一些熱門的研究主題是，利用資料倉儲技術建立資料倉庫，以及對資料倉庫的資料進行線上分析處理，以期找出有用的資訊，支援決策活

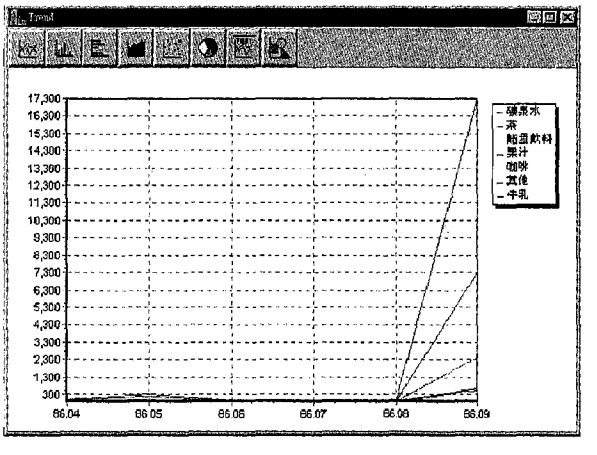

圖6：各類非碳酸飲料銷售趨勢

動。在資料倉儲方面，資料倉庫大部分是多維度資料庫，有不少研究探討塑模多維度資料庫，提出多維度資料模式。(Chaudhuri 1997) 談到大部分的資料倉儲使用星狀綱要 (star schema)，有部分使用雪花綱要 (snowflake schema)。星狀綱要的維度資料表格沒有正規劃化，有更正異常 (update anomalies) 的問題，但較方便瀏覽維度資料。雪花綱要的維度資料經過正規劃化，雖然較容易維護，但查詢指令往往需要較複雜的合併 (join) 運算。

(Cabibbo 1997) 提出 MultiDimensional 資料模式，一個 MultiDimensional 資料庫綱要包括一組維度綱要 (dimensions) 及一組事實綱要 (f-table schemes)。一個維度綱要爲一格構 (lattice)，定義維度資料的分類架構，如產品維度綱要。事實資料表格根據事實綱要儲存原始資料，包括至少一個數值型態的欄位。(Gyssens 1996) 將一資料表格的屬性分成參數 (parameters) 及度量 (measures) 兩類。多個參數屬性可集合成一維度。文中提出運算子可直接對維度做查詢。此模式並沒儲存類別資訊，無法對資料表格進行一般化處理。

基本上，大部分的多維度資料塑模研究，將多維度資料庫的資料分成兩大類：維度資料及事實資料，提出利用維度綱要及事實綱要塑模維度資料及事實資料。然而，固定式維度綱要缺乏彈性，不易調整，必須將分析系統離線由專業的系統開發人員修改資料庫綱要，然後再重建維度資料。因此，無法充分支援線上沙盤推演式的決策模式，本研究中，我們提出無綱要式的屬性階層，塑模維度資料，分析人員可以直接線上調整維度資料。

在線上分析處理，以及強化資料庫管理系統對資料分析的功能方面，(Red 1995) 提出 Red Brick System，增強結構化查詢語言的群聚功能，以及一些新的循序函數，有效率地提升處理順序相關的運算，包括排名 (ranking)、移動平均、移動加總等。往上合成及往下展開是資料分析非常重要的功能，(Gray 1997) 以資料方塊 (data cube) 爲基礎，提出 cube 運算子，分析人員可以進行合成與展開資料分析。在 (Kimball 1995) 中，作者指出結構化查詢語言並不足以直接支援各類的商業資料的分析。弱點之一是無法以簡潔的指令進行兩段時序性數據的比較，然而時序性數據的比較，是分析商業資料的一項重要功能。例如，比較去年營業額及今年的營業額。文中作者提出以關鍵字 ALTERNATE 擴充結構化查詢語言，解決數據比對問題。(Chatziantoniou 1996) 以一些很直覺的查詢需求，卻很難以簡潔的結構化查詢語言表示的範例，說明傳統結構化查詢語言不足之處。並且針對此弱點，作者提出一個新的關聯代數運算子間 C 飪 B 算子雖然沒有增加關聯代數的表達能力，卻大幅簡化部分聚集查詢的表示法，同時也讓執行最佳化變得較容易。(Agrawal 1995) 提出形狀定義語言 SDL，能夠在大量歷史資料中，分析數值資料，找出這些資料所隱含的資訊。SDL 允許使用者自行定義趨勢的形狀，對歷史資料進行近似查詢。例如，查詢歷史資料中有連續五個月銷售業績小幅上升，同時有連續三個月大幅下降的區段。

柒、結論

大量資料中蘊藏寶貴的資訊，適當的分析及應用，可以支援企業的決策活動。近年來，有不少研究探討利用資料倉儲技術建立資料倉庫，然後進行線上分析處理，以強化資料庫管理系統對決策活動的支援。在塑模多維度資料倉庫的資料模式，大部分研究提出維度綱要表示維度分類架構，然而此法缺乏彈性，不易調整。本論文提出無綱要式的屬性階層，其優點

爲 (1) 容易塑模複雜的維度資料：維度資料的分類，可以不需要都是相同的分類深度。再者，沒有複雜的維度綱要不容易命名的問題。(2) 屬性階層可隨分析的需要而動態調整，無須拘泥於維度綱要。我們認爲無綱要式的屬性階層具有高度彈性，對於沙盤推演式的決策模式能提供較好的支援。此外，配合擴充的屬性階層，我們擴充結構化查詢語言，以便充分利用儲存在屬性階層的資料。GENERALIZE 指令及 PARENT 函數，可以進行高層次資料查詢。在數值資料的分析方面，TREND 函數可以進行趨勢分析及異常分析。

我們目前將 GENRALIZE 指令、PARENT 函數及 TREND 函數建構在現有的關聯式資料庫管理系統上，利用第三代程式語言，處理掉擴充的部分，再呼叫傳統的結構化查詢語言。一個未來的研究方向是，將這些擴充的指令整合到資料庫管理系統的查詢處理引擎，直接做查詢處理的最佳化，以便進一步提升高層次查詢及資料分析的速度。

致謝

本研究承蒙國科會補助研究經費（計劃編號：NSC 88-2416-H-224-020），特此致謝。另外，感謝洪銘鋒及劉世富等人協助開發雛形系統。

參考文獻

1. Agrawal, R., G. Psaila, E. Wimmers and M. Zait, "Querying Shapes of Histories", Proc.of the 21st VLDB Conf., Zurich, Switzerland 1995
2. Codd, E. F., Providing OLAP (on-line analytical processing) to user-analysis: An IT madate, Technical Report, E. F. Codd and Associates, 1993.
3. Cabibbo, L. and R. Torlone, "Querying Multidimensional Databases", Proc. 6th DBPL Workshop, 1997, pp. 253-269.
4. Chatziantoniou, D. and K. A. Ross, "Querying Multiple Features of Groups in Relational Databases", Proc. of the 22nd VLDB Conf., Mumbai(Bombay), India, 1996
5. Chaudhuri, S. and K. Shim, "Including Group-by in Query Optimization", Proc. of Int'l Conf. on Very Large Data Bases, 1994, pp. 354-366
6. Chaudhuri, S. and K. Shim, "Optimizing Queries with Aggregate Views", Proc. of Conf. on Extending Database Technology, 1996, pp.167-182
7. Chaudhuri, S. and U. Dayal, "An Overview of Data Warehousing and OLAP Technology", ACM SIGMOD RECORD, Vol 26, No. 1, March 1997, pp.65-74
8. Chaudhuri, S. and K. Shim, "An Overview of Cost-based Optimization of Queries with Aggregates", IEEE Data Engineering Bulletin, Sept. 1995.
9. Elmasri, R. and S. B. Navathe, Funcdamntals of Database Systems, Benjamin/Cummings Publishing Company, Inc., 1994, pp.491-525.
10. Finkelstein, R., "MDD: Database Reaches the Next Dimension", Database Programming And Design, April 1995, pp.27-38.
11. Graefe, G., "Query Evaluation Techniques for large Databases", ACM Computing Surveys 25(2), June 1993, pp.73-170.
12. Gray, J., S. Chaudhuri, A. Layman, D. Richart and M. Venkatrao, "Data Cube:

A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals", Data Mining and Knowledge Discovery, March 1997, pp.29-53

13. Gupta, A., V. Harinarayan and D. Quass, "Aggregate-query Processing in Data Warehousing Environments", Proc. of Int'l Conf. on Very Large Data Bases, 1995, pp. 358-369.
14. Gyssens, M. and L. Lakshmanan, "A Foundation for Multi-Dimensional Databases", Proc. of the 22nd VLDB Conf. 1996.
15. Inmon, W., Building the Data Warehouse, John Wiley, 1996.
16. IS 9075 International Standard Database Language SQL, document ISO/IEC 9075:1992
17. Kimball, R. and K. Strehlo, "Why Decision Support FAILS and How To FIX It", ACM SIGMOD Record, Vol. 24, No. 3, September 1995
18. Kimball, R. The Data Warehouse Toolkit, John Wiley, 1996.
19. Klug, A., Equivalence of Relationa Algebra and Relational Calculus Query Languages Having Aggregate Functions", JACM, 29(3), July 1982.
20. Li, C. and X. Wang, "A Data Model for Supporting On-Line Analytical Processing", Proc. of conf. on Information and Knowledge Management, Nov. 1996, pp.81-88.
21. Muralikrishna, M., "Improved Unnesting Algorithms for Jon and Aggregate SQL Queries", Proc. of Int'l Conf. on Very Large Data Bases, 1992.
22. Red Brick Systems White paper, Decision-Makers, Business Data and RI-SQL, RedBrick Systems, Los Gatos, CA, 1995
23. Widom, J. "Research Problems in Data Warehousing", Proc. of 4th Int'l. Conf. on Information and Knowledge Management, 1995
24. Yan. W. and P. Larson, "Performing Group-by before Join", Proc. of Int'l Conf. on Data Engineering, 1994, pp.89-100.
25. Yan W. and P. Larson, "Eager Aggregation and Lazy Aggregation", Proc. of Int'l Conf. on Very Large Data Bases, 1995, pp. 345-357.