IBM Informix Warehouse Accelerator

效能就是一切

2011 年 3 月

技術白皮書

作者：Keshava Murthy

IBM Informix Development 資深技術主管

想像一下，貴公司 C 級主管拿到分析師的報表，是在幾秒鐘內，而不是好幾個小時。想像一下，分析與洞察的速度不斷增加。想像一下，持續提升業務的執行速度，讓許多資料點在幾秒鐘內即可處理完畢。想像一下，持續提升貴公司的倉儲查詢效能，而不需要持續監視與調整系統。想像一下，執行這些作業，卻不用建立 Cube、彙總表格、索引、統計資料或建立分割策略。

在 Informix Warehouse Accelerator 的延伸測試結束之後，美國 Skechers 的資深資料庫架構設計人員 Ashutosh Khunte 表示：「在使用 Informix Warehouse Accelerator 之前，對於隨處都有超過一百萬個資料列的企業倉儲而言，執行複雜的庫存與銷售分析查詢要花費數分鐘到 45 分鐘。當我們使用 Informix Warehouse Accelerator 時，同樣的查詢只要 2 到 4 秒即可完成！這表示這軟體的執行速度快了 60 到 1400 倍，平均加速係數大於 450—整個過程不需要任何的索引，或是建立 Cube、調整查詢或變更應用程式！」

Advanced Data Tools 的 Lester Knutsen (Informix Data 冠軍)從他的客戶美國農業部拿到資料並執行他的工作。他表示：「Informix Warehouse Accelerator 提供的查詢效能令人驚豔，執行速度比過去快了 30 倍。列狀技術(columnar technology)可以節省許多處理時間，減少 9.5 小時到 15 分鐘的工作量，整個執行過程都不需要進行任何的資料庫調整，或是對實體儲存體進行管理。」

ATG IT Consulting GmbH 的 Thomas Gemesi 表示：「Informix Warehouse Accelerator (IWA)提供在單一平台上進行資料倉儲(Data warehousing)及線上交易處理(Online Transaction Processing，OLTP)的功能。查詢僅需幾秒鐘即可執行完畢，而且不需要把錢投資在任何其他的工具。」

改變是常態，對速度的需求也是如此。您必須瞭解模式(預測趨勢)，然後調整貴公司的業務流程。資料分析可協助您瞭解業務的趨勢與轉變。比任何人更早、持續不斷，並以較低的總持有成本進行資料分析，將可為您帶來勝過競爭者的優勢。

Informix Warehouse Accelerator 提供極致的效能，並免除大部分傳統資料倉儲需要進行的調整。本軟體是專為在數秒內處理大量資料而設計。本軟體專為企業設計，在需要的時候可以提供正確的資料，而不會讓企業的維護成本增加，或是讓應用程式基礎建設改變。

創新是關鍵。處理器到記憶體的存取速度，和處理器到磁碟的存取速度，兩者間的效能瓶頸會隨著時間增加。傳統的資料庫系統採用少量的記憶體組態，並且透過作業最佳化將磁碟 IO 降到最低，並使用緩衝區將最近存取的資料保存在記憶體。下滑的記憶體價格和可以輕鬆擁有 TB 記憶體容量的系統需要全新的設計。IBM 讓這些系統支援 3TB 的記憶體，而且這個容量預計還會增加。加速程式利用這種趨勢，在記憶體內壓縮和快取所有資料並消除磁碟 IO。每次推出的新版處理器，都會加入更多的核心和更大的快取記憶體。加速程式利用這種趨勢，將同步處理平行化與最小化，並將用於晶片快取上的演算法最佳化。這種做法讓查詢效能達到新高，而設計則走向樸實雅致。

單純的力量

窮完美之道，乃減而有損，非添而有盈。
-- Antoine de Saint-Exupery

加速程式具有三個重要的原則：

- 每個工作負載的加速不需要手動進行調整
- 支援現有的企業工具及應用程式
- 搭配現有的倉儲基礎建設使用

如果您的環境已經擁有 Informix，加速程式會與您現有的環境互相銜接。如果是全新部署，有多種平台可供您彈性選擇，並且可以和 Informix 資料庫伺服器自由整合。

只要將資料交給加速程式，加速程式隨時都可以展現其峰值效能。

「不需要」建立索引、不需要索引顧問、不需要索引重組。加速程式的查詢處理引擎會用邏輯的方式，在數毫秒或數秒內掃描數百萬個資料列。深度列狀資料表示法、壓縮資料的查詢處理、利用新式處理器功能的創新演算法，讓您不再需要索引。

「不需要」收集統計資料。傳統的最佳化工具要建立更好的查詢規畫，必須依靠定期收集統計資料。有了加速程式之後，系統會自動判斷結合順序，並持續使用星狀結合的規畫。缺少索引反而可以簡化選項，而執行階段最佳化設定(將於本文稍後說明)可免除對收集統計資料與統計資料收集顧問的需求。

「不需要」建立分割(片段)架構。資料會自動進行垂直與水平分割。因為資料格採用深度列狀儲存體的架構，所以查詢也可以從垂直與水平分割區剪除(也稱為片段忽略)而獲益。如此可免除資料表分割架構的規畫。

「不需要」針對每個查詢或工作負載進行調整。在安裝加速程式的期間，您提供基本的記憶體與儲存體組態。一致的規畫、消除磁碟 IO、快速掃描與結合可免除執行階段的調整。

「不需要」管理儲存體。所有的資料都儲存在記憶體內，而磁碟中僅儲存一份記憶體內映像(in-memory image)的副本。您不需要為表格(索引)規畫及建立儲存體空間。

「不需要」昂貴的硬體。加速程式是專為在平價量產硬體(commodity hardware)上執行，並隨著您的基礎建設發展所設計。加速程式可以在 Linux/Intel 的電腦上執行，並與 Linux/Intel、AIX/Power、HP-UX/Itanium、Solaris/Sparc 上的 Informix 資料庫伺服器整合。

「不需要」變更資料庫。加速程式會利用您資料倉儲中現有的邏輯綱目。

「不需要」變更應用程式。加速程式會連接到 Informix 資料庫伺服器作為資源。Informix 資料庫伺服器會知道儲存在加速程式中的資料集市，並將相關查詢自動引導至加速程式。您不需要變更應用程式或工具。

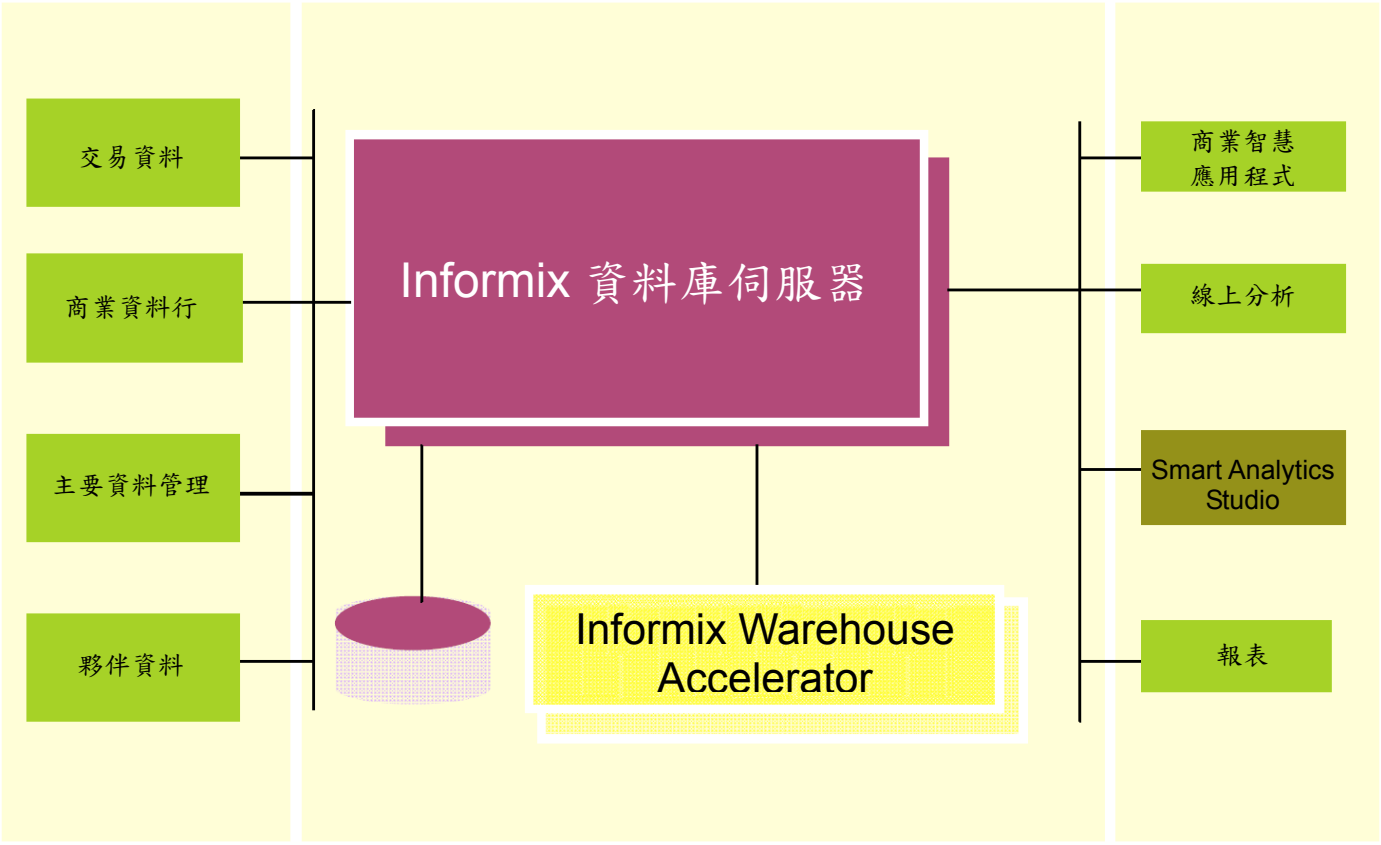
「不需要」建立彙總表格或具體化檢視。加速程式表格掃描與結合至少比傳統的資料庫快了一個數量級。如此可免除彙總表格的建立與維護，以及相關顧問的使用。

「不需要」設定分頁或區塊大小。深度列狀技術會自動判斷與最佳化記憶體內資料格的大小。

「不需要」暫存空間的配置或監視。系統也會將中間結果集壓縮並使用較少的空間。

「不需要」查詢及最佳化工具提示即可加速查詢。最佳化工具會持續使用星狀結合的規畫。加速程式查詢處理器會根據執行階段的統計資料調整順序。

Informix Warehouse 的架構



Informix 資料庫伺服器是一種具有可擴充與高可用性的資料庫伺服器，可以讓數以萬計的使用者推動重要的交易任務與分析應用程式。Informix 資料庫伺服器是一個完整的倉儲資料庫伺服器，內含 ETL 工具、對時間週期資料管理的內建支援、線上作業、深度壓縮、查詢最佳化，以及專為複雜資料倉儲工作設計的處理技術。Informix 的客戶可以使用 Cognos 或 Microstrategy 等商業智慧工具來分析並改善其業務。

一般而言，大家會說資料倉儲查詢很複雜有其原因。這些查詢會存取數百萬和數十億個資料列，其中間結果相當龐大，資料列為平行排列時的查詢效能最佳。星狀結合最佳化的技術，是專為處理傳統系統中的查詢而設計。這一切都可以讓經驗豐富的資料庫管理員(DBA)瞭解工作負載，並調整出適當的系統參數與索引。

Informix Warehouse Accelerator 提供超越以往的倉儲查詢效能，免去資料庫調整的工作，而且不需要對您現有的應用程式與工具環境進行任何的變更。本白皮書的其餘部分，會說明加速程式的技術、用法，以及和 Informix 的整合。

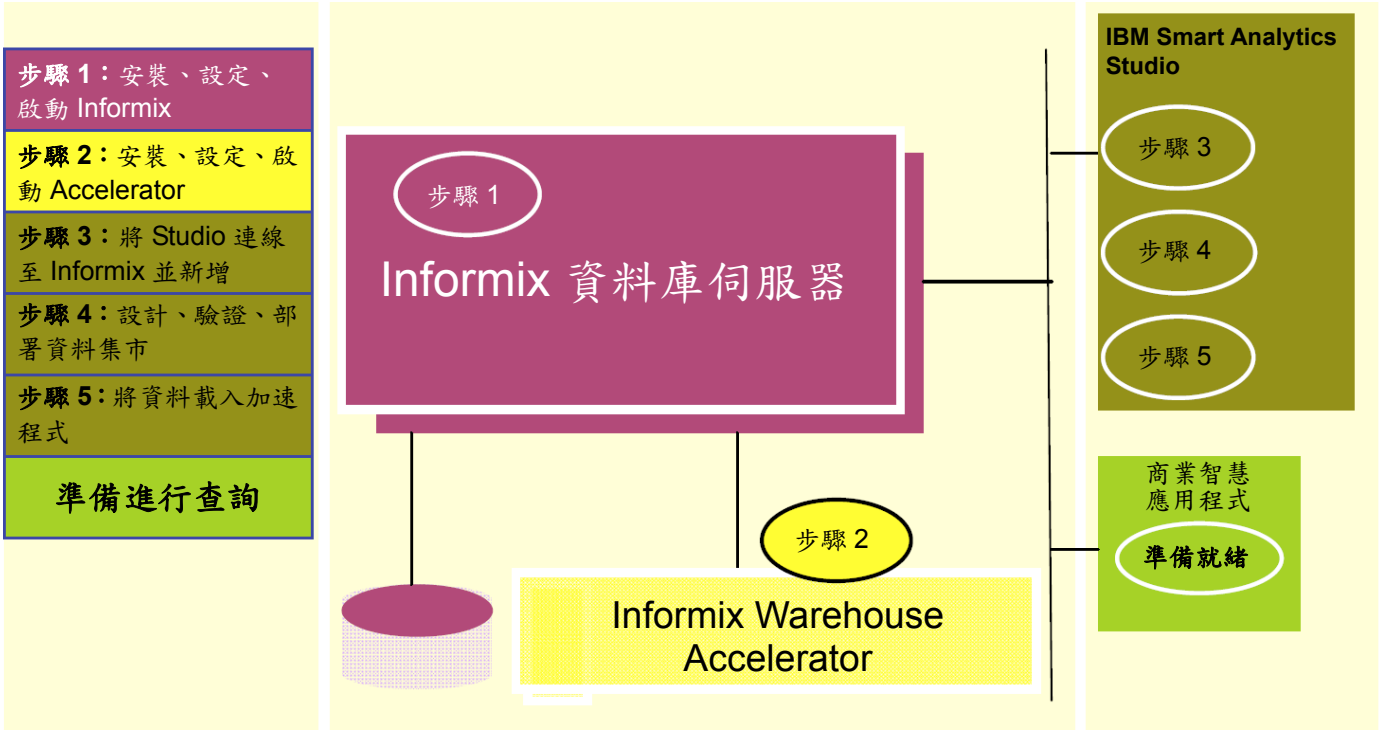
部署 Informix Warehouse Accelerator

Informix Warehouse Accelerator 始終可以在 Intel 伺服器上的高效能 Linux 作業系統中執行。Informix 資料庫伺服器和加速程式可以在相同或不同的伺服器上執行。在 Intel 伺服器中，您可以在 Linux/Intel、AIX/ Power、HPUX/HP Itanium、Solaris/Sparc 上執行 Informix 資料庫伺服器，然後在不同的 Linux 上執行加速程式。

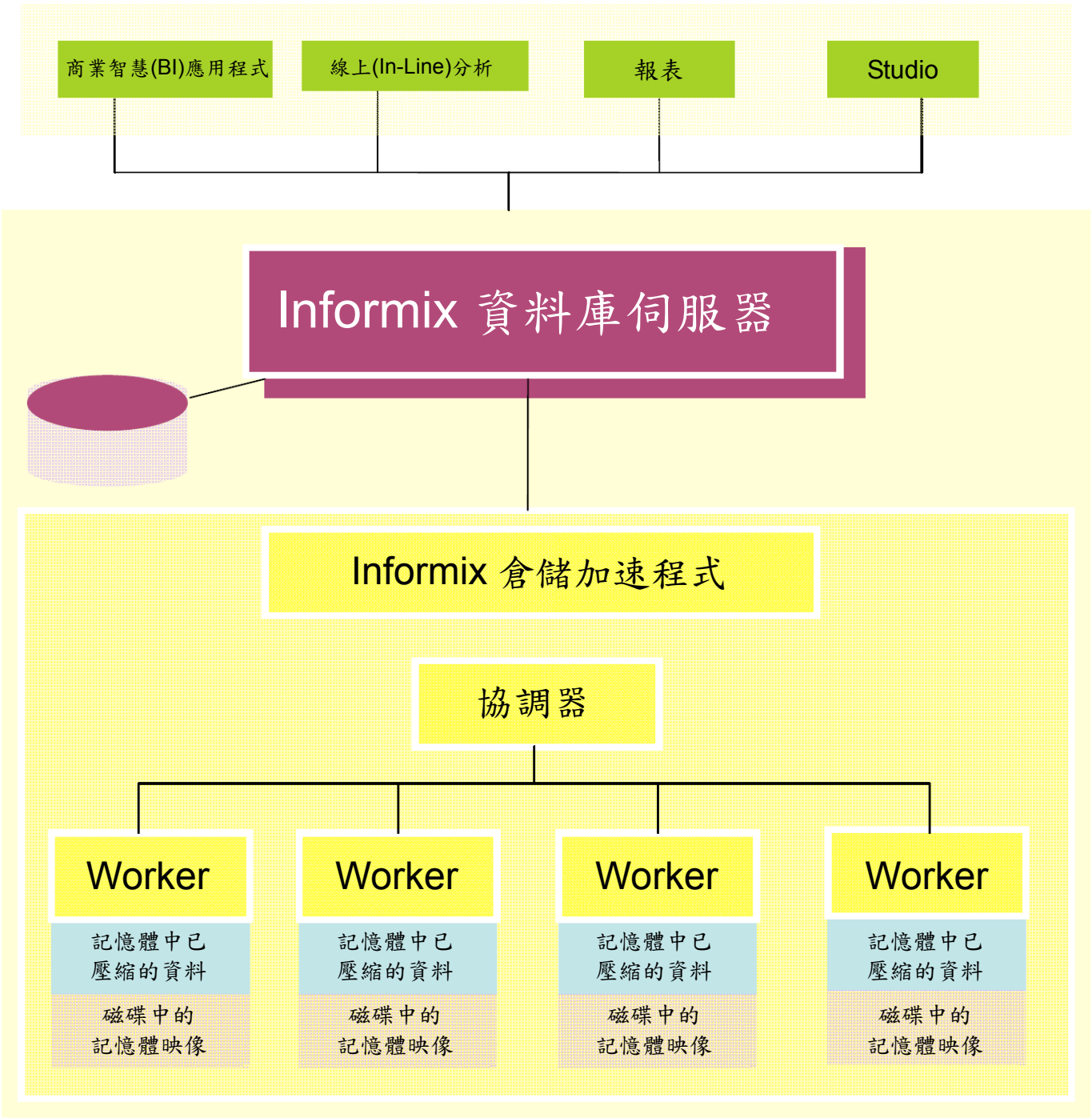
加速程式是 Informix Ultimate Warehouse Edition (IUWE)的一部份，其中包含 Informix 資料庫伺服器(Ultimate Edition)和 Informix Warehouse Accelerator。加速程式媒體包含加速程式二進位檔，以及用於設定和管理包含 Informix 資料庫伺服器的 IBM Smart Analytics Optimizer Studio 工具。以下是從零開始部署的作業順序。本產品包含快速入門手冊和管理指南，內有安裝與設定 Informix 及加速程式的完整使用說明。您將會發現，加速程式僅需要一些簡單的設定步驟：用於資料備份的檔案系統位置、記憶體數量和 CPU 資源。設定的相關資訊將於稍後在本文件及管理指南中討論。

IBM Smart Analytics Optimizer Studio 是一種以 Eclipse 為基礎的工具，可以用來管理加速程式，並且可以用來定義與部署從 Informix 到加速程式的資料集市。Studio 分為兩種版本：一種適用於 Linux，另一種則適用於 Windows。您可以在同一部電腦上使用 Linux 版本的 Informix 作為 Informix 資料庫伺服器，或是選擇在另一部電腦上使用。如果您要使用 Windows 版本的 Informix，請將自我解壓縮二進位檔傳送到使用 Windows 的電腦上，然後進行安裝。

部署 Informix 資料庫伺服器與加速程式的步驟。



Informix Warehouse Accelerator 元件



加速程式可以透過 TCP/IP 網路連線到 Informix 資料庫伺服器。如果加速程式與資料庫伺服器位於同一部電腦上，兩者便可透過 TCP/IP 迴路連線進行通訊。加速程式是由協調器與工作處理程序所組成。Informix 透過協調器處理程序和加速程式進行通訊。協調器與工作處理程序共同承擔查詢處理的責任。您指定的組態會判斷這些處理程序的數量，以及使用的記憶體數量與 CPU 資源。

在 Informix 和加速程式執行[請參閱手冊取得詳細資料]之後，透過 Studio 新增加速程式資訊。順利完成此步驟後，Informix 會將加速程式連線資訊加入其 SQLHOSTS 檔案。加速程式的 SQLHOSTS 檔案項目看起來會像是這樣：

```
sales_acc     group          -- c=1,a=4b3f3f457d5f552b613b4c587551362d2776496f226e714d75217e22614742677b424224
sales_acc_1   dwsoctcp          127.0.0.1          21022 g=sales_acc
```

加速程式的名稱為 sales_acc。Informix 會使用該名稱建立新的群組。十六進位的數值，只是用來確認 Informix 資料庫與加速程式(sales_acc)通訊的驗證碼。協調器的名稱為 sales_acc_1。資料庫通訊協定 DW 是用於透過 TCP/IP 進行通訊。資料庫通訊協定非常接近針對 Informix 和加速程式通訊最佳化的 DRDA 通訊協定。127.0.0.1 (TPC/IP 迴路位址)表示加速程式和 Informix 正在同一部電腦上執行。有了大量的 Worker，您將有多個協調器處理失效接手。在這些組態中，每個協調器都會對應一個項目。

協調器具有三個重要的功能。

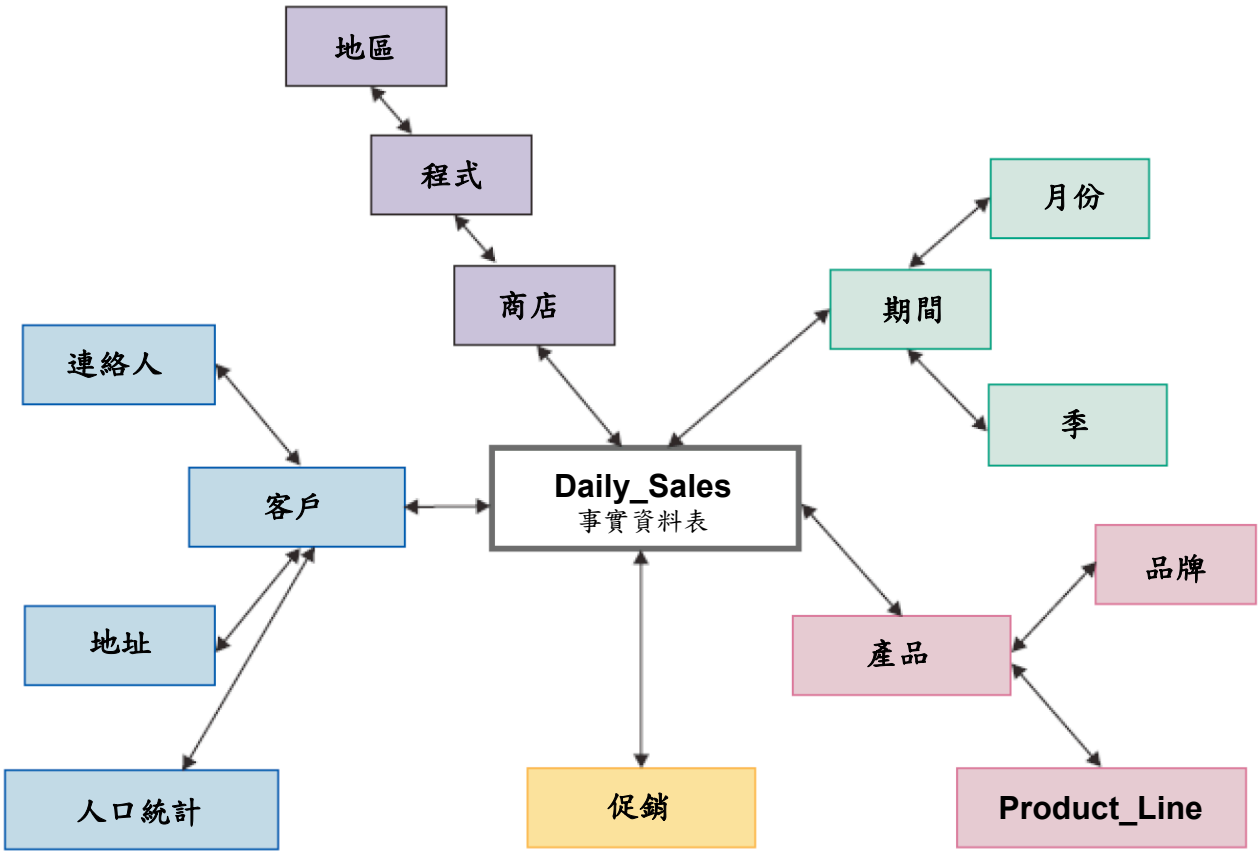
1. 這是 Informix 資料庫伺服器通訊的重點。資料庫伺服器會連線至協調器以分送資料，協調器也是傳送查詢和擷取結果集的主要連絡人。
2. 在進行資料讀取的階段中(步驟 8)，協調器會在多個 Worker 間散發資料。接著協調器會收集整個壓縮字典、將其合併並重新散發字典，因此所有人都是使用相同的參考字典。
3. 在查詢處理的階段中，協調器會接收來自資料庫伺服器的查詢、將查詢傳送至每個 Worker、取得中間結果集、合併群組後解壓縮資料，並在將最後的結果傳送至 Informix 資料庫伺服器前，視需要排序資料。

工作處理具有兩個重要的功能。

1. 在資料讀取的階段中，每個 Worker 會利用頻率分割分析傳入的資料、將資料自動垂直和水平分割到資料格，並利用「深度列狀(Deep Columnar)」技術壓縮資料。資料壓縮完畢之後，資料會寫入磁碟供復原使用。我們將於本白皮書稍後再詳細說明深度列狀的技術。
2. 工作處理的第二個功能為查詢處理壓縮的資料。每個 Worker 會維護維度表格和部分事實表格的壓縮副本。每個 Worker 會將中間結果傳回至協調器。記憶體內的查詢處理已經 100%完成。

協調器與 Worker 一起運作，可以平行化所有查詢的執行並快速傳回結果。

Wikipedia 將資料集市(Data Mart)定義為資料倉儲的子集，通常會導向特定的業務線或團隊。我們將在本白皮書中使用此定義。您在 Informix 中的企業資料倉儲可能包含來自銷售、庫存、客戶服務及市場資料的全部內容。除此之外，您會加速提供價值非常高的資料集市。例如，業務經理可能想要分析銷售與庫存資料，以瞭解市場趨勢並建立適當的促銷方法。因此您只需要使用銷售與庫存事實表，即可加速分析所需要的資料集市。在加速程式的內容中，每個資料集市包含一或多個雪花狀網目(snowflake schema)。每個雪花狀網目擁有一個事實表格和相關的維度表格。在下列的雪花狀網目中，DAILY_SALES 是一個事實表格，而且和描述事實的維度相關。



以下的章節將說明從設計資料集市到使用加速程式的步驟。這些步驟參照「部署 Informix Warehouse Accelerator」一節中的圖表。

使用 IBM Smart Analytics Optimizer Studio 設計與部署資料集市。資料集市會定義事實和維度表格，以及兩者間的關係。維度表格內含的資料列，通常明顯較事實表格(例如：生產資訊或客戶資訊)少。在某些情況下，維度表格可能相當龐大(例如：加州所有的居民)。一旦您找出要建立資料集市的表格，您就必須定義事實和所有維度表格間的關係。

資料集市的驗證步驟，可以確定已定義表格間的所有關係。請在確定此步驟中的任何錯誤都已解決之後再部署資料集市。在部署的步驟中，IBM Smart analytics Optimizer Studio 會將 XML 格式的資料集市定義傳送至會傳回 SQL 中定義的加速程式。此定義會儲存為 Informix 系統型錄中的「檢視」，其中包含特殊旗號與相關資訊。這種檢視稱為「加速查詢表格(Accelerated Query Table，AQT)」。此 AQT 稍後將用來比對查詢，並將相符的查詢重新導向至加速程式。

以下是 AQT 的範例，內含做為事實表格與其維度的 daily_sales：產品、商店、客戶、促銷。這個 AQT 會自動建立並供您參考。使用者或資料庫管理員不會使用此檢視。來自應用程式或工具的 SQL 會以平常的方式，利用綱目中定義的表格運作。

```
create view "dwa"."aqt2dbca0d9-509d-434b-9cc9-4a12c6de6b3d"
("COL16","COL17","COL18","COL19","COL20","COL21","COL22","COL23","COL24","COL25","COL2
6","COL27","COL28","COL29","COL30","COL31","COL32","COL33","COL34","COL35","COL36","CO
L37","COL38","COL39","COL40","COL41","COL42","COL43","COL44","COL45","COL46","COL47","
COL07","COL08","COL09","COL10","COL11","COL12","COL13","COL14","COL15","COL48","COL49"
,"COL50","COL51","COL52","COL53","COL54","COL55","COL56","COL57","COL58","COL59","COL6
0","COL61","COL01","COL02","COL03","COL04","COL05","COL06","COL62","COL63","COL64","CO
L65","COL66","COL67","COL68","COL69","COL70","COL71","COL72","COL73","COL74","COL75","
COL76","COL77","COL78","COL79","COL80","COL81") as
  select x0.perkey ,x0.storekey ,x0.custkey ,x0.prodkey ,x0.promokey
    ,x0.quantity_sold ,x0.extended_price ,x0.extended_cost ,x0.shelf_location
    ,x0.shelf_number ,x0.start_shelf_date ,x0.shelf_height ,x0.shelf_width
    ,x0.shelf_depth ,x0.shelf_cost ,x0.shelf_cost_pct_of_sale
    ,x0.bin_number ,x0.product_per_bin ,x0.start_bin_date ,x0.bin_height
    ,x0.bin_width ,x0.bin_depth ,x0.bin_cost ,x0.bin_cost_pct_of_sale
    ,x0.trans_number ,x0.handling_charge ,x0.upc ,x0.shipping
    ,x0.tax ,x0.percent_discount ,x0.total_display_cost ,x0.total_discount
    ,x1.perkey ,x1.calendar_date ,x1.day_of_week ,x1.week ,x1.period
    ,x1."year" ,x1.holiday_flag ,x1.week_ending_date ,x1."month"
     ,x2.prodkey ,x2.upc_number ,x2.package_type ,x2.flavor ,
    x2.form ,x2.category ,x2.sub_category ,x2.case_pack ,x2.package_size
    ,x2.item_desc ,x2.p_price ,x2.category_desc ,x2.p_cost ,x2.sub_category_desc
    ,x3.storekey ,x3.store_number ,x3.city ,x3.state ,x3.district
    ,x3.region ,x4.custkey ,x4."name" ,x4."address" ,x4.c_city
    ,x4.c_state ,x4.zip ,x4.phone ,x4.age_level ,x4.age_level_desc
```

```
    ,x4.income_level ,x4.income_level_desc ,x4.marital_status
    ,x4.gender ,x4.discount ,x5.promokey ,x5.promotype ,x5.promodesc
    ,x5.promovalue ,x5.promovalue2 ,x5.promo_cost
from
 (((((("informix".daily_sales x0 left join "informix".period x1 on (x0.perkey
    = x1.perkey ) )left join "informix".product x2 on (x0.prodkey
    = x2.prodkey ) )left join "informix"."store" x3 on (x0.storekey
    = x3.storekey ) )left join "informix".customer x4 on (x0.custkey
    = x4.custkey ) )left join "informix".promotion x5 on (x0.promokey
    = x5.promokey ) );
```

在**讀取**的步驟中，來自資料庫伺服器表格的資料 Snapshot 會傳送至加速程式。在這個階段中，加速程式會將資料散發給它的每個 Worker。Worker 會分析常見數值的資料以及直欄間的關係，然後垂直和水平分割資料。選擇分割之後，系統會使用本白皮書稍後說明的深度列狀處理程序壓縮資料。請注意，在這個階段中，資料會壓縮並保留在記憶體，並在磁碟中保留一份壓縮資料的副本。這個階段不會建立索引，也不會建立彙總表格或 Cube。您可以從 Informix 資料庫伺服器定期重新整理這些資料(例如：每晚)。

一旦讀取完畢，您就可以開始使用加速程式中的資料集市。只要將資料交給加速程式，立刻就可以開始感受 Informix 的峰值效能。Informix 倉儲加速程式包含用於設計、驗證、部署和讀取的命令行公用程式。這個公用程式對於寫入讓本處理程序自動化的程式碼相當實用。

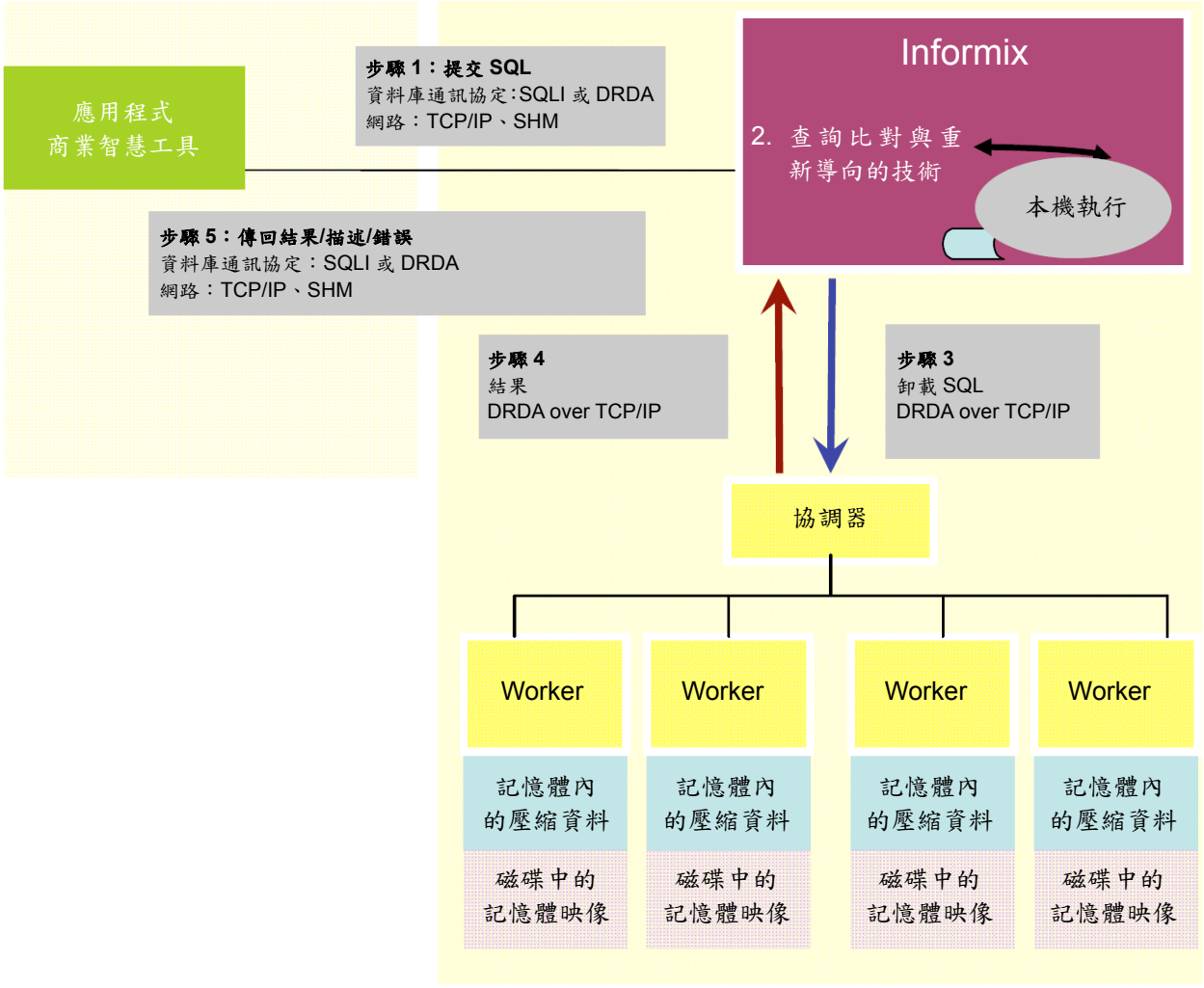
資料的**查詢**，通常與事實表格和一或多個維度互相結合，然後尋找資料中的特定模式相關。在下列範例中，web_sales 和 4 個其他的維度互相結合。如果您已建立並部署包含這些表格的資料集市，Informix 會將查詢和特定資料集市的定義檢視(加速查詢表格)進行比對，然後將查詢傳送至加速程式。這種作法，就像從一個 Informix 資料庫伺服器傳送到另一個伺服器的分散式查詢。查詢的結果會透過相同的連線傳回，然後傳回至用戶端應用程式。

```
select first 100 i_item_id,
        avg(ws_quantity) avg_quantity,
        avg(ws_list_price) avg_list_price,
        avg(ws_coupon_amt) avg_coupton_amt,
        sum(ws_sales_price) sum_sales_price
 from web_sales, customer_demographics, date_dim, item, promotion
 where ws_sold_date_sk = d_date_sk and
       ws_item_sk = i_item_sk and
       ws_bill_cdemo_sk = cd_demo_sk and
       ws_promo_sk = p_promo_sk and
       cd_gender = 'F' and
```

```
        cd_marital_status = 'M' and
        cd_education_status = 'College' and
        (p_channel_email = 'N' or p_channel_event = 'N') and
        d_year = 2001
group by i_item_id;
order by sum(ws_sales_price) desc;
```

加速程式會要求查詢透過零或多個維度和事實表格互相結合，並且利用定義資料集市的關係時指定的結合鍵(join key)和每個表格互相結合。在主控端(dominant side)中，加速程式也支援具有事實表格的內部結合(inner join)和左結合(left join)。請參閱「管理指南」取得查詢資格的詳細資料。加速程式會按先後順序執行每筆查詢，期間不會有任何中斷。所有的資料和中間結果都位於記憶體內。每個 Worker 都有要與其結合的維度副本。每個結合執行緒會嘗試將雜湊表格(hash table)快取至 L2 快取，以將記憶體的存取最佳化。工作處理會以少量的資料交換進行個別掃描與結合，並與其他的 Worker 進行同步處理。和傳統的系統花費數分鐘和數小時相比，每筆查詢通常都會在數秒內完成。因此，加速程式會將查詢要求排入佇列，並逐筆執行查詢。

使用加速程式的查詢流程



設定注意事項

在安裝 Informix Warehouse Accelerator 的期間，系統會要求您設定節點的數量(協調器、Worker)、供 Worker 使用的記憶體，以及協調器節點的記憶體組態。系統會自動判斷協調器與 Worker 的數量。例如，如果您定義 5 個節點，系統會自動建立 1 個協調器與 4 個 Worker 節點。「管理指南」內有詳細說明。請試想 4 個 Worker 和 1 個協調器的組態。舉例來說，您部署一個資料集市，內含桌子銷售的事實表格，以及客戶、商店和時間的維度。系統會壓縮客戶、商店和時間的維度表格，並將其保留在每個 Worker 的專用記憶體內。因此，記憶體內會有 4 個維度表格的副本，而銷售的事實表格資料列則會平均分給 4 個 Worker。每個 Worker 會在主記憶體內維護客戶、商店和時間的維度表格，以及事實表格內 25%的資料列－即本案例中的銷售。

假設有足夠的 CPU 容量，資料傳輸率通常會隨著 Worker 的數量增加。大量的 Worker 確實有助於提高查詢處理的速度，但效果並不明顯。Worker 數量對查詢的影響，也取決於查詢的內容。

如果是這樣，您應該為每個 Worker 配置多少記憶體？系統需要記憶體做哪些事情？

一般來說，記憶體中未壓縮的 Informix 資料和已壓縮的 Informix Warehouse Accelerator 資料，我們已經知道兩者的壓縮比是 3 比 1。如果銷售的事實表格和客戶、商店、時間的維度表格，兩者個總和約為 100GB，而其中有大部分是由而銷售表格使用，您將會需要約 33GB 的記憶體來儲存資料。利用 Open Admin Tool (OTA)或直接查詢型錄，即可輕鬆找出表格的大小。

每個 Worker 都需要充分的執行階段記憶體，才能在執行階段儲存中間結果。Worker 會動態配置及釋放動態查詢處理所需的記憶體。此處的規畫與 Informix 中的暫存空間規畫類似。您預期工作負載會排序多少中間結果？與您資料的關聯程度，以及每個種類中有多少群組？雖然很難確實回答，但是我們發現，另外 1/5 到 1/3 的資料大小，通常就已經足夠使用。

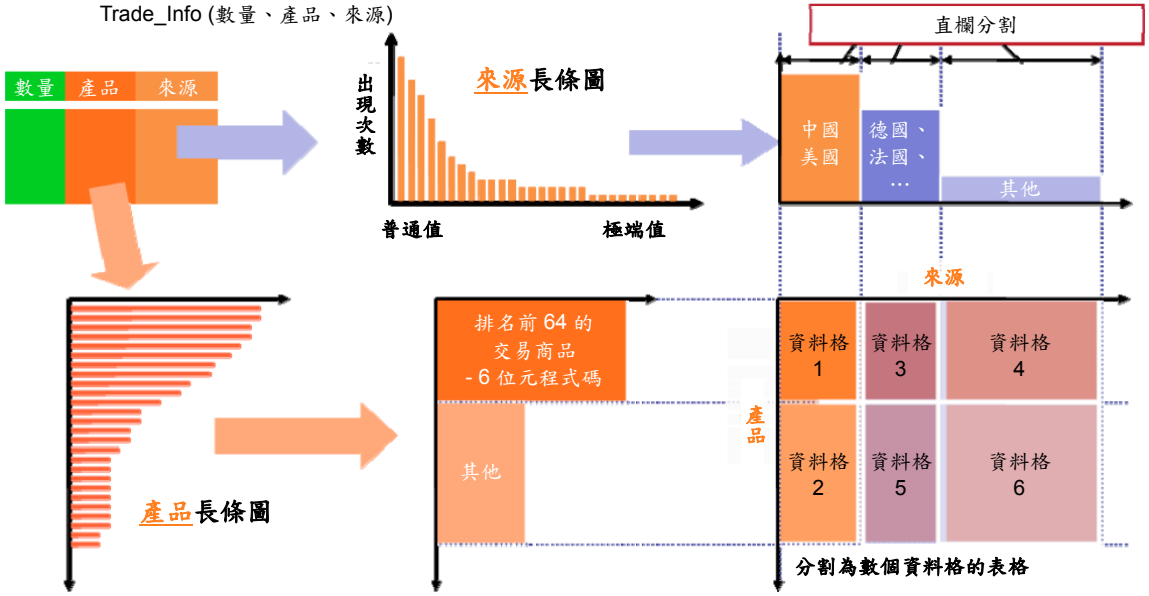
加速程式查詢處理的最後階段會在協調器完成。協調器需要記憶體以供合併、解壓縮和排序結果集使用。再次強調，這裡所需要的記憶體，取決於您預期的結果集大小。請注意，當您發出內含 FIRST 子句，例如 “SELECT FIRST 1000… ORDER BY sum(sales.amount)” 的查詢時，協調器就必須排序所有的資料，才能取得前 1000 筆資料。當然，一旦協調器建立前 1000 個群組，即可取代或拒絕新的群組。

峰值效能

加速程式的極致效能，是集 IBM 研究和開發團隊開發之技術於大成的結果。本白皮書提供加速程式用來提高效能，以及免除調整與維護工作的技術與技巧概要。列於參考文獻章節的論文，提供更多基本理論與技巧的詳細資料。這些論文已發表在知名的學術期刊或研討會。IBM 已提出這些技巧的專利申請。

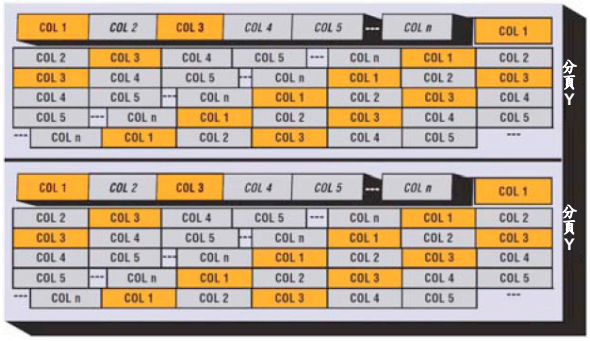
深度列狀的技術勝過傳統的列狀儲存體。用於壓縮資料的極致壓縮與查詢處理，可以免除查詢處理期間的磁碟 IO，並允許大量的記憶體內倉儲。利用多核心的架構和 SIMD 技術，不需要索引或彙總表格，即可讓您見識到令人驚艷的處理速度。現在就讓我們來了解各項技術。

頻率分割



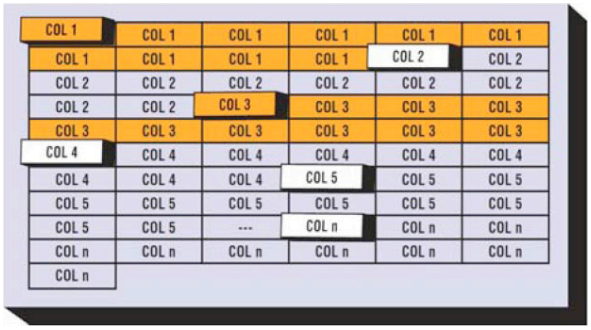
系統會針對直欄與相關直欄群組中的常見值分析資料集市中的每個表格，以決定最適合結合成 Tuplet 的直欄。在上述的範例中，兩個直欄的產品和來源會互相關聯，也因此會合併形成 Tuplet。Tuplet 是元組(Tuple)的片段，而元組是完整的資料列。Huffman 編碼(Huffman encoding)的優點，在於大多數的常見值會以最小的位元組數量進行編碼。在上述的圖表中，系統會使用 Huffman 編碼，將排名前 64 的產品值和來源(美國、中國)中的最常見值合併成較小的資料格 1。這種技術會提高壓縮的效率，而且可以用來評估相等和範圍述詞。因為查詢處理是在已壓縮的資料上完成，故位元組變少後速度會加快。

加速程式中的列狀儲存體

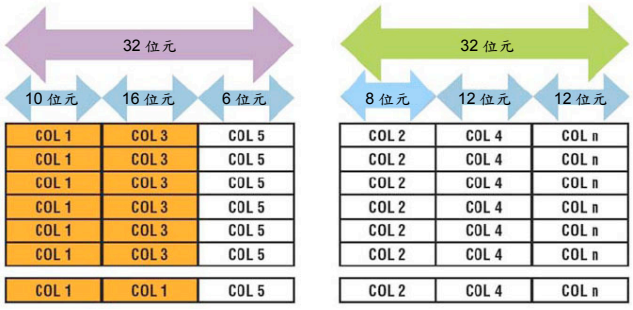


傳統的**逐行**資料庫會將完整的資料列(也稱為元組)儲存在一個分頁中，後面接著另一個資料列。假設查詢與大多數的直欄值有關，這種設計就可以讓資料列的 IO 效率最佳化。如果資料是壓縮在儲存體中，資料列就不會在每次提取時都進行壓縮。這對每筆查詢存取少量資料列的傳統工作負載而言，是一種有效率的方法。

在分析查詢中，您通常會存取數百萬或數十億個資料列，但每次查詢都會分析事實表格中直欄子集間的關係。例如，您可能對 2010 年每個地區的項目銷售總額感興趣。進行這項查詢，只需要存取事實表格中的三個直欄：項目、銷售總額和地區，以及與維度的結合。在這個案例中，存取並解壓縮每個資料列的做法效率不佳。



列狀資料庫會將每個直欄值儲存在一起。當您每次插入或讀取資料時，系統會解壓縮每個資料列來分隔直欄值，而當您每次存取資料列時，系統會分別提取直欄值，然後將這些直欄值壓縮成資料列。因為直欄列儲存在一起，所以您可以獲得較好的壓縮。在上述的範例中，我們看到分析查詢通常和資料列的子集有關。如果要在列狀儲存體資料庫中進行這項查詢，您只能提取儲存項目、銷售與地區的分頁。對於會存取資料列的大型子集並執行循序掃描的查詢，列狀儲存體可以提高其效率。



Informix Warehouse Accelerator 會將資料儲存在**直欄群組**，或是表格的垂直分割(稱為「Bank」)。載入每個 Bank 的資料列片段(或是元組)則稱為「Tuplet」。指派給 Bank 的直欄稱為特定資料格，因為直欄的長度會因為不同的資料格而有所不同。

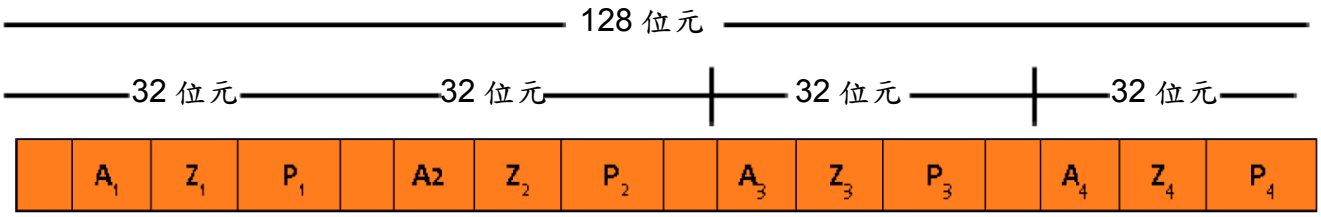
指派會根據直欄是否載入 Bank 使用裝箱演算法(bin-packing algorithm)，其中直欄的寬度是一個字組的部分片段，而不是工作負載的使用量。此外，掃描需要存取的 Bank，只有那些包含任何特定查詢參照的直欄，而不需要掃描那些包含查詢沒有參照的直欄。這種縱向選取和純直欄儲存體最小化磁碟 IO 的方式類似。加速程式會將所有的資料儲存在記憶體內，所以不會有磁碟 IO。即使如此，Informix Warehouse Accelerator 也沒有磁碟 IO，這種技術會讓要掃描的記憶體數量降到最低，並節省相當可觀的 CPU 循環。

單一指令多重資料平行處理原則

請看以下的查詢：

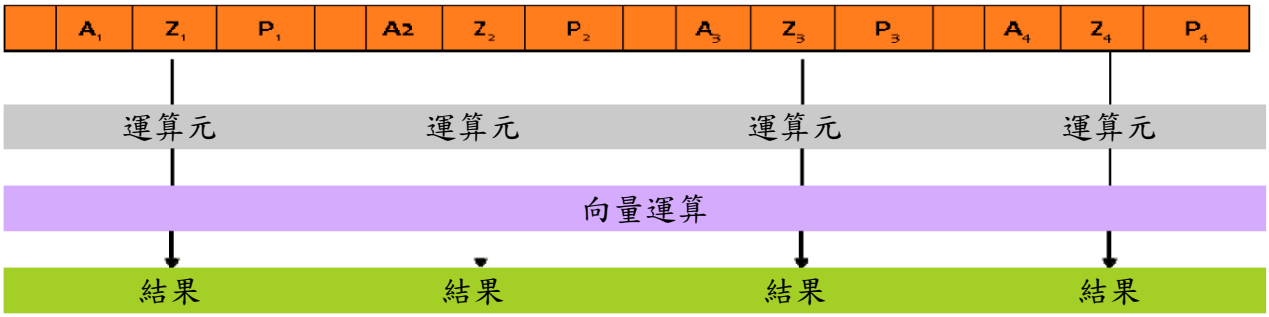
```
SELECT SUM(s.amount) FROM sales AS s WHERE s.prid = 100 GROUP BY s.zip;
```

如果(A)、prid (P)、zip(Z)間的直欄來自相同的 Bank，其中的多個值即可同時載入 128 CPU 登錄。在本案例中，我們可以同時載入 12 個直欄值。



Intel Xeon 處理器上的 SIMD 指令，可以在 128 位元的登錄(register)上運作。加速程式中的壓縮技術，通常需要數個位元組供每個直欄使用，因此可以讀取每個 128 位元登錄中的許多欄位。我們可以讀取多個值，然後將述詞同時套用在所有的直欄上。在進行查詢處理的期間，這種超載作業會發生在所有已配置的核心上，因而造成查詢的極端並行化。

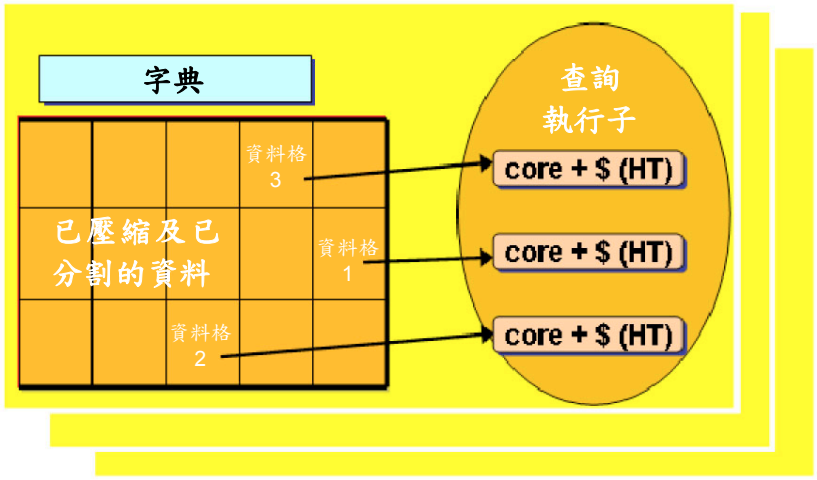
一旦出現這種情況，您就可以同時對所有的 12 個數值使用單一 CPU 指令進行運算，如此將可顯著增加效能。



查詢處理

到目前為止，我們已經探討了資料的編碼和儲存方式，以及在 Micro Level 完成查詢處理的方式。現在讓我們探索 Macro Level 的查詢處理。

有效的掃描提供查詢處理的基礎。我們知道資料會組織成 Tuplet、Bank 和資料格。對掃描來說，資料格是處理的單位。加速程式會動態產生適當的執行緒以利用設定的 CPU 資源，並將資料格指派給每個核心。這種在已壓縮的資料上使用 SIMD 指令進行的掃描作業，會利用 Huffman 編碼的優點。述詞評估「GROUP BY」會在已壓縮的資料上完成，但彙總則是在解壓縮的資料上完成。這種編碼技術的進行，也和非常密集的雜湊函數相同，會允許處理器 L2 快取中雜湊表格的快取，以及進行快速查詢。



這種編碼技術可以在已壓縮的資料上用非常快的速度執行 GROUP BY。結合兩個密集編碼的雜湊表格，會造成疏鬆的值集。加速程式會偵測這些狀況，並動態更換成線性探測。這些技術的結合，讓查詢處理可以在已壓縮的資料上完成。

我們知道資料集市是一個星狀或雪花狀網目，內含大型的事實表格和各種維度表格。查詢會在事實到維度和維度到其他維度間，利用結合述詞將事實表格和各種維度表格互相結合。每個 Worker 會針對每筆查詢為表格建立一個雪花狀模型。接著，Worker 會從分支最靠近外部的邊緣開始，順著分支的方向往事實表格內進行處理。Worker 會處理每個雪花狀的分支，而其結果會用來作為到下一層的維度輸入。首先，系統會將本端述詞套用至維度表格，以建立完整鍵(qualified key)的清單。這些來自維度表格的完整鍵，會和事實表格(或是作為雪花狀分支上事實的維度表格)結合成下一層的彙總與關聯。系統會遞迴套用這個程序，直到每個 Worker 處理完整個結合。因為每個 Worker 具有維度表格的副本，所以在進行查詢處理的期間，Worker 間會有少量的資料交換，而且每個 Worker 可能會以最高的效能繼續進行處理。接著 Worker 會將中間結果傳送至協調器。

因為只有一個資料表示-壓縮列狀表格資料-所以加速程式每次都會對每個表格沿用相同的程式碼路徑。資料格的所有效率和查詢的區塊消除都來自壓縮編碼。基本上，加速程式不需要考量索引和彙總表格，每次差不多都會沿用相同的處理程序，因此加速程式的效能也會前後一致。

協調器節點會從每個 Worker 取得結果、將中間結果合併到適當的群組、解壓縮資料，然後在將資料透過 DRDA 通訊協定傳送至 Informix 資料庫伺服器之前執行 ORDER BY 及 HAVING 指令。接著 Informix 資料庫伺服器會將資料經過路由傳送至應用程式。

結論

Informix Warehouse Accelerator 針對複雜查詢處理的創新方法，可以透過更快提供答案的方式提高貴公司的生產力，而不會增加您的工作或是讓您的預算超支。因為 Informix Warehouse Accelerator 和 Informix 資料庫伺服器緊密整合，所以可以讓您利用這種環境，並視需要分割 Informix 和加速程式間的負載。

更快的反應時間表示可以更快得到答案、更快速地洞察，以及用更快的速度調整業務。您可以規畫加速倉儲高價值的部分，並不斷發展基礎結構以滿足業務的各項需求。

進一步的資訊

如需進一步了解 Informix Warehouse Accelerator 及 Informix Ultimate Warehouse Edition，請聯絡您的 IBM 業務代表或 IBM 事業夥伴，或造訪下列網頁：
http://www.ibm.com/informix
http://www.ibm.com/informix/warehouse

延伸閱讀

- **VLDB 2008**：Lin Qiao、Vijayshankar Raman、Frederick Reiss、Peter Haas、Guy Lohman，*《與多核心 CPU 共用的主記憶體掃描》(Main-Memory Scan Sharing for Multi-core CPUs)*
- **VLDB 2008**：Ryan Johnson、Vijayshankar Raman、Richard Sidle、Garret Swart，*《逐行平行述詞評估》(Row-Wise Parallel Predicate Evaluation)*
- **VLDB2006**：Vijayshankar Raman、Garret Swart，*《如何扭乾表格：Entropy 壓縮的關係與查詢壓縮的關係》(How to wring a table Dry: Entropy Compression of Relations and Querying Compressed Relations)*
- **SIGMOD 2007**：Allison L. Holloway、Vijayshankar Raman、Garret Swart、David J. DeWitt，*《如何用位元換取時間子：資料庫掃描中壓縮與頻寬的權衡》(How to barter bits for chronons: compression and bandwidth trade offs for database scans)*
- **ICDE 2008**：Vijayshankar Raman、Garret Swart、Lin Qiao、Frederick Reiss、Vijay Dialani、Donald Kossmann、Inderpal Narang、Richard Sidle，*《常數時間的查詢處理》 (Constant-time Query Processing)*
- **BTW 2009**：Knut Stolze、Vijayshankar Raman、Richard Sidle、Oliver Draese，*《讓 BLINK 更接近功能完全發揮的 SQL》(Bringing BLINK Closer to the Full Power of SQL)*

銘謝

本產品是由 IBM Almaden Research、IBM Böblingen Lab 及 IBM Informix 團隊合作開發。感謝 Informix 團隊檢閱並改進本白皮書的內容。

IBM Corporation
Software Group
Route 100
Somers, NY 10589
U.S.A.

於台灣列印
2011 年 3 月