

(19) 日本国特許庁(JP)

(12) 公表特許公報(A)

(11) 特許出願公表番号

特表2004-534981

(P2004-534981A)

(43) 公表日 平成16年11月18日(2004. 11. 18)

(51) Int.Cl.⁷

G06F 17/30

F I

G06F 17/30

4 1 4 Z

G06F 17/30

1 8 0 D

G06F 17/30

4 1 9 B

テーマコード (参考)

5 B 0 7 5

審査請求 未請求 予備審査請求 有 (全 339 頁)

(21) 出願番号	特願2002-547041 (P2002-547041)	(71) 出願人	503193878
(86) (22) 出願日	平成13年11月29日 (2001. 11. 29)		ラファイエット・ソフトウェア・インコーポレーテッド
(85) 翻訳文提出日	平成15年5月28日 (2003. 5. 28)		アメリカ合衆国・カリフォルニア・9 4 3
(86) 国際出願番号	PCT/IB2001/002792		0 6 - 2 1 5 5 ・ パロ ・ アルト ・ エル ・ カ
(87) 国際公開番号	W02002/044943		ミノ ・ リアル ・ 3 0 0 0 ・ ファイヴ ・ パロ
(87) 国際公開日	平成14年6月6日 (2002. 6. 6)		・ アルト ・ スクウェア
(31) 優先権主張番号	00403329. 6	(74) 代理人	100064908
(32) 優先日	平成12年11月29日 (2000. 11. 29)		弁理士 志賀 正武
(33) 優先権主張国	欧州特許庁 (EP)	(74) 代理人	100108578
(31) 優先権主張番号	00403330. 4		弁理士 高橋 詔男
(32) 優先日	平成12年11月29日 (2000. 11. 29)	(74) 代理人	100089037
(33) 優先権主張国	欧州特許庁 (EP)		弁理士 渡邊 隆
(31) 優先権主張番号	00403331. 2	(74) 代理人	100101465
(32) 優先日	平成12年11月29日 (2000. 11. 29)		弁理士 青山 正和
(33) 優先権主張国	欧州特許庁 (EP)		

最終頁に続く

(54) 【発明の名称】 データベース・システムでデータを編成し、問合せを処理する方法、およびそのような方法を実施するためのデータベース・システムおよびソフトウェア製品

(57) 【要約】

参照テーブルは、データ属性に関連付けた列、およびデータの集まりでその属性に割り当てられた関連ワードを含む行を有し、このワードは、独立した数のレコードを有する異なるデータ・テーブルに由来する。記憶されているデータは、属性に関連付けたワード・シソーラス、およびそれぞれシソーラス項目に関連付けた参照テーブル行識別子を含む。属性に関連付けた各ワード・シソーラスは、データ集合体の中のこのデータ属性に割り当てられたワードごとに個々の項目を有する。参照テーブルは、仮想テーブルでもよいが、全シソーラスの項目用に統一された代数的枠組を定義する。

【特許請求の範囲】

【請求項 1】

データベース・システムで情報を編成する方法であって、属性のグループが定義され、データ集合体のワードを前記属性に割り当て、属性のグループを、それぞれが個々のデータ・テーブルに関連付けられた複数のサブグループに分割し、各データ・テーブルが、関連するサブグループの各属性のための列、および関連するサブグループの属性に割り当てられた少なくとも 1 つのワードを備えるデータ・テーブル・レコードを含むための行を有し、データ・テーブル・レコード間にリンクが定義され、各リンクが、目標テーブルと、それぞれが前記目標テーブルのレコードを指定するリンク値を含むリンク列を有する対応ソース・テーブルとを有し、それにより前記リンク値がそれぞれ、前記リンク値を含むソース・テーブルのレコードと前記リンク値によって指定された目標テーブルのレコードとの間のリンクを表し、

— 個々の識別子をデータ・グラフに割り当てるステップを含み、各データ・グラフが、それぞれ前記グループの属性に指定された関連属性値を表し、データ・グラフの各属性値が、デフォルト値か、または前記データ集合体のワードであり、各データ・グラフのワードが、連結されたデータ・テーブル・レコードに由来し、さらに、

— それぞれ前記グループの属性に関連付けられた複数のワード・シソーラスを記憶するステップを含み、データ集合の属性に少なくとも 1 回は指定されたワードごとに前記属性に関連付けられたワード・シソーラスが、前記ワードを含む別個の項目を有し、さらに、

— それぞれがワード・シソーラスの項目に関連付けられたデータ・グラフ識別子リストを表すデータを記憶するステップを含み、属性に指定されたワードに関するシソーラス項目に関連付けられたデータ・グラフ識別子が、前記属性に指定された前記ワードを有するデータ・グラフに割り当てられた識別子を含む方法。

【請求項 2】

前記グループの属性に関連付けられた各ワード・シソーラスの項目を前記属性に指定されたワードに基づいてソートする、請求項 1 に記載の方法。

【請求項 3】

さらに、それぞれがデータ・グラフに関連付けられた複数の行、およびそれぞれが属性サブグループに関連する複数の列を有するリンク・テーブルを記憶するステップを含み、リンク・テーブルの各行が、それぞれの列に、前記行に関連付けられて、前記列の 1 つに関連するサブグループの属性に指定されたデータ・グラフで表される各属性値がデフォルト値であることを示す値か、または前記行に関連付けられたデータ・グラフで表現され、前記列の 1 つに関連するサブグループの属性に割り当てられたデータ集合の少なくとも 1 つの記憶ワードを検索するためのリンク値を含む、請求項 1 または 2 に記載の方法。

【請求項 4】

前記データ・テーブルが記憶され、前記サブグループに関連付けられたリンク・テーブルの列に含まれる各リンク値が、前記サブグループに関連付けられたデータ・テーブルの列を識別するデータを備える、請求項 3 に記載の方法。

【請求項 5】

リンク・テーブルが、1 つの属性で構成されたサブグループに関連付けられた少なくとも 1 つの列を有し、前記列に含まれる各リンク値が、データ集合の前記 1 つの属性に指定されたワードである、請求項 3 または 4 に記載の方法。

【請求項 6】

リンク・テーブルが、1 つの属性で構成されたサブグループに関連付けられた少なくとも 1 つの列を有し、前記列に含まれる各リンク値が、前記 1 つの属性に関連付けられたワード・シソーラスの項目へのポインタである、請求項 3 に記載の方法。

【請求項 7】

データ・グラフの少なくとも 1 つにデフォルト値が指定されたグループの属性に関連付けられた各ワード・シソーラスが、さらに、デフォルト値の項目を有し、これによって前記データ・グラフ識別子リストの 1 つが、デフォルト値の前記シソーラス項目と関連付けら

れ、前記属性に指定された前記デフォルト値を有するデータ・グラフに割り当てられた識別子を含む、先行する請求項のいずれか1項に記載の方法。

【請求項8】

識別子をデータ・グラフに割り当てるステップが、

- ー 対応ソース・テーブルのリンク列で指定されていない任意の目標テーブル・レコードについて、リンク列に前記目標テーブル行を指定するリンク値および各属性列にデフォルト値を有する前記ソース・テーブルの追加行にダミー・レコードを挿入することを含み、任意の目標テーブルの任意のレコードが、対応ソース・テーブルのリンク列で指定されるまで、このようなダミー・レコードの挿入が繰り返され、さらに、
- ー 空白リストを表すように、データ・グラフ識別子リストを表すデータを初期化することと、
- ー どのリンクの目標テーブルでもない少なくとも1つの根テーブルのレコードを処理することとを含み、各根テーブル・レコードの処理が、
- ー 前記根テーブル・レコードの関連属性値、および前記レコードにあるリンクの内容に基づいて連続的に決定された他のデータ・テーブルの対応するレコードで構成された個別データ・グラフを構築することと、
- ー 使用可能な識別子を前記データ・グラフに割り当てることと、
- ー 属性ごとに、前記データ・グラフの前記属性に指定された値に関する関連ワード・シソーラスの項目を選択し、前記データ・グラフに割り当てられた識別子を前記リストに追加するため、前記選択項目に関連付けられたデータ・グラフ識別子を表すデータを更新することとを含む、請求項7に記載の方法。

【請求項9】

各根テーブル・レコードを処理することが、さらに、

- ー 個々にデータ・テーブルに関連付けられた複数の列を有するリンク・テーブルの行を作成することを含み、リンク・テーブルの前記行が、データ・テーブルに関連付けられた各列に、データ・グラフが前記データ・テーブルに関してダミー・レコードから構築されたことを示す値か、またはデータ・グラフに表示された属性値が記憶されている前記データ・テーブルの行を指定するリンク値を含み、
- 前記リンク・テーブルが、データ・グラフに表示されたデータ集合体のワードをデータ・テーブルから検索するために使用される、請求項8に記載の方法。

【請求項10】

さらに、ダミー・レコードをデータ・テーブルから削除するステップを含む、請求項9に記載の方法。

【請求項11】

さらに、リンク列をデータ・テーブルから削除するステップを含む、請求項9または10に記載の方法。

【請求項12】

さらに、前記グループの属性に、および前記属性のワード長より短い接頭語長に関連付けられ、前記接頭語長を有する接頭語値ごとに個々の項目を有し、データ集合体にある前記属性に少なくとも1回指定された少なくとも1ワードの対応接頭語と一致する少なくとも1つのマクロワード・シソーラスを記憶するステップを含み、前記マクロワード・シソーラスの項目は、前記属性に指定され、前記接頭語値と一致する対応接頭語を有するワードを有するデータ・グラフに割り当てられた各識別子を含むデータ・グラフ識別子リストに関連付けられる、先行する請求項のいずれか1項に記載の方法。

【請求項13】

少なくとも1つの属性が、異なる接頭語長に関連付けられ、記憶された複数のマクロワード・シソーラスを有する、請求項12に記載の方法。

【請求項14】

属性に関連付けられた各マクロワード・シソーラスの項目が、接頭語値に基づいて記憶され、前記属性に関連付けられたワード・シソーラスの項目が、前記属性に関連付けられた

ワードに基づいて記憶される、請求項 1 2 または 1 3 に記載の方法。

【請求項 1 5】

少なくとも 1 つの属性が、前記属性のワード長より短い個々の接頭語長に関連付けられ、記憶された Q 個のマクロワード・シソーラスを有し、 Q がプラスの整数であり、前記 Q 個のマクロワード・シソーラスがそれぞれ、 $1 \leq q \leq Q$ になるようにシソーラス・レベル・パラメータ q を有し、接頭語長が、レベル・パラメータ q の減少関数であり、前記属性に関連付けられたワード・シソーラスがレベル・パラメータ $q = 0$ を有し、レベル・パラメータ $q \geq 1$ を有する各マクロワード・シソーラスが、レベル q の接頭語値のために設けた項目ごとに、前記レベル q 接頭語値と一致する対応接頭語を有する最低または最高レベル $q - 1$ のマクロワードまたはワードに対応するレベル $q - 1$ のシソーラスの項目を指定するデータを含む、請求項 1 4 に記載の方法。 10

【請求項 1 6】

少なくとも 1 つのシソーラス項目が、前記項目に関連付けられたデータ・グラフ識別子リストを表すコード化データが記憶されている少なくとも 1 つのメモリ領域を指すためのデータを備える、先行する請求項のいずれか 1 項に記載の方法。

【請求項 1 7】

前記メモリ領域が、個別に前記シソーラス項目に割り当てられたファイルを備える、請求項 1 6 に記載の方法。

【請求項 1 8】

少なくとも 1 つのメモリ領域を指す前記データが、前記シソーラス項目を設けるための属性値を備え、前記属性値が、前記シソーラス項目に割り当てられた各ファイルにアクセスするために使用する個別ファイル名の一部である、請求項 1 7 に記載の方法。 20

【請求項 1 9】

前記メモリ領域が、それぞれが次アドレス・フィールドを有する複数のレコードを有するデータ・コンテナの一部を備え、前記一部が、前記データ・コンテナのレコード・チェーンと定義され、チェーンが、次アドレス・フィールドによって定義され、少なくとも 1 つのメモリ領域を指す前記データが、前記データ・コンテナにあるチェーンの第 1 のレコードの個別アドレスを備える、請求項 1 6 に記載の方法。

【請求項 2 0】

さらに、各チェーンのレコードが連続アドレスを有するように、データ・コンテナに記憶されたレコードをグループ分けするステップを含む、請求項 1 9 に記載の方法。 30

【請求項 2 1】

前記データ・コンテナが、個別にシソーラスに割り当てられる、請求項 1 9 または 2 0 に記載の方法。

【請求項 2 2】

前記データ・コンテナが、複数のシソーラスで共用される、請求項 1 9 または 2 0 に記載の方法。

【請求項 2 3】

前記グループの属性に関連付けられたシソーラスが、前記属性に割り当てられたワードに基づいてシソーラス項目を記憶した指標レジスタを有し、各項目が、補助テーブルの行を指すワード索引を含み、補助テーブルの各行が、チェーンの第 1 のレコードのデータ・コンテナにアドレスを含む、請求項 1 9 から 2 2 のいずれか 1 項に記載の方法。 40

【請求項 2 4】

少なくとも 1 つのメモリ領域を指す前記データが、前記データ・コンテナにあるチェーンの最終レコードの個別アドレスを備える、請求項 1 9 から 2 3 のいずれか 1 項に記載の方法。

【請求項 2 5】

データ・グラフ識別子リストを表す前記コード化データが、個々に前記リストの識別子に等しい整数を備える、請求項 1 6 から 2 4 のいずれか 1 項に記載の方法。

【請求項 2 6】

データ・グラフ識別子リストを表すために複数のフォーマットを設け、各シソーラス項目が、自身に関連付けられたデータ・グラフ識別子リストを表すために使用するフォーマットの表示を含む、先行する請求項のいずれか1項に記載の方法。

【請求項27】

データ・グラフ識別子リストを表す前記データが、少なくとも1つのシソーラス項目に、 n 個の連続するコード化層を有するコード化方式によって獲得されたコード化データを備え、 n が少なくとも1に等しい数であり、各層が、前記層の入力リストの整数をカバーする範囲をサブセットに分割する所定のパターンを有し、前記識別子リストが、前記シソーラス項目の第1の層の入力リストであり、最終層以外の層について、入力リストの少なくとも1つの整数を含む各サブセットの位置を、前記層のパターンで表す整数リストが、次層の入力リストを形成し、前記コード化データが、層ごとに、および入力リストの少なくとも1つの整数を含むサブセットごとに、前記サブセットにおける入力リストの各整数の位置を表すデータと、少なくとも前記層が最終層である場合に、前記サブセットの位置を前記層のパターンで表すデータとを備える、先行する請求項のいずれか1項に記載の方法。

【請求項28】

コード化データが、個別にシソーラス項目に割り当てられたファイルを含む複数のファイルに記憶される、請求項27に記載の方法。

【請求項29】

前記シソーラスの項目に関するコード化データを含むため、コード化データが、個々のシソーラスに割り当てられた少なくとも1つのファイルを含む複数のファイルに記憶される、請求項27または28に記載の方法。

【請求項30】

前記複数のシソーラスの項目に関するコード化データを含めるため、コード化データが、複数のシソーラスに割り当てられた少なくとも1つのファイルに記憶される、請求項27から29のいずれか1項に記載の方法。

【請求項31】

各層のパターンは、整数のサブセットが同じ数の整数で構成される連続インターバルであるようなパターンである、請求項27から30のいずれか1項に記載の方法。

【請求項32】

前記整数の数が、層ごとに2の整数累乗である、請求項31に記載の方法。

【請求項33】

サブセットにおける各層の入力リストの各整数の位置を表すデータが、前記整数が前記層の入力リストに属するか否かを示すため、サブセットの個々の整数に割り当てられた各ビットを含むビットマップ・セグメントで構成される、請求項27から32のいずれか1項に記載の方法。

【請求項34】

層 n のパターンにおける各サブセットの位置が、前記サブセットにおける層 n の入力リストの整数の位置を表すデータに関連付けられたコード化データに含まれる整数ランクによって表される、請求項27から33のいずれか1項に記載の方法。

【請求項35】

各層のパターンにおけるサブセットの位置が、前記サブセットにおける前記層の入力リストの整数の位置を表すデータに関連付けられた前記層のコード化データに含まれる整数ランクによって表される、請求項27から33のいずれか1項に記載の方法。

【請求項36】

$n \geq 2$ であり、それぞれ複数のレコードを有する層 k のデータ・コンテナが、 $1 \leq k \leq n$ の場合にコンピュータ・メモリに提供され、層 k のデータ・コンテナの各レコードが、サブセットの位置を層 k パターンで表す層 k の整数ランクに割り当てられ、サブセットの位置を層 k パターンで表す層 k ランクに関連付けられた層 k のデータ・コンテナの各レコードが、データ・グラフ識別子リストに関する層 k 入力リストの任意の整数の前記サブセッ

トにおける位置を検索するデータを含むため、第1のフィールドを有し、それにより前記層 k ランクと、前記第1のフィールドに含まれたデータから検索可能な任意の位置との組合せが、 $k > 1$ の場合は、層 $k - 1$ データ・コンテナの個々のレコードが関連付けられた層 $k - 1$ のランクを、 $k = 1$ の場合は、前記データ・グラフ識別子リストの識別子を決定する、請求項 27 から 35 のいずれか1項に記載の方法。

【請求項 37】

$1 \leq k \leq n$ の場合、サブセットにおける層 k 入力リストの任意の整数の位置を検索するため、層 k データ・コンテナのレコードの第1のフィールドに含まれる前記データが、前記整数が前記層 k 入力リストに属するか示すため、前記サブセットの個々の整数に関連付けられた各ビットを含むビットマップ・セグメントを備える、請求項 36 に記載の方法。

10

【請求項 38】

$1 \leq k \leq n$ の場合、層 k ランクに関連付けられた層 k データ・コンテナの各レコードが、さらに、前記層 k ランクを含むための第2のフィールドを有する、請求項 37 に記載の方法。

【請求項 39】

各データ・コンテナが、前記データ・コンテナのレコードの第1および第2のフィールドがそれぞれ記憶されている少なくとも2つのファイルを備え、前記ファイルが別個にアクセス可能である、請求項 38 に記載の方法。

【請求項 40】

$1 \leq k \leq n$ の場合、層 k データ・コンテナの各レコードが、さらに、層 $k + 1$ パターンのサブセットにおける層 $k + 1$ 入力リストの整数の位置を表す数字を含むために第2のフィールドを有し、

20

$1 \leq k \leq n$ の場合、層 k パターンのサブセットにおける層 k 入力リストの任意の整数の位置を検索するために、層 k ランクと関連付けられた層 k データ・コンテナのレコードの第1のフィールドに含まれた前記データが、層 k パターンの前記サブセットにおける前記層 k 入力リストの整数の位置を表す数字を含む第2のフィールドを含む層 $k - 1$ データ・コンテナの少なくとも1つのレコードを指すポインタを備え、それにより層 $k - 1$ データ・コンテナの前記レコードが、前記層 k ランクと前記数字によって表された位置との組合せによって決定された層 $k - 1$ ランクに関連付けられる、請求項 36 に記載の方法。

【請求項 41】

30

サブセットにおけるデータ・グラフ識別子リストの任意の整数の位置を検索するために、層1データ・コンテナのレコードの第1のフィールドに含まれる前記データが、前記整数が前記リストのデータ・グラフ識別子を表すか示すため、前記サブセットの個々の整数に関連付けられた各ビットを含むビットマップ・セグメントを備える、請求項 40 に記載の方法。

【請求項 42】

層 n ランクに関連付けられた層 n のデータ・コンテナの各レコードが、さらに、前記層 n ランクを含むために第2のフィールドを有する、請求項 40 または 41 に記載の方法。

【請求項 43】

$1 \leq k < n$ の場合に、各層 k データ・コンテナが、前記データ・コンテナのレコードの第1および第2のフィールドが個々に記憶された少なくとも2つのファイルを備え、前記ファイルが別個にアクセス可能である、請求項 40 から 42 のいずれか1項に記載の方法。

40

【請求項 44】

$1 \leq k \leq n$ の場合、層 k データ・コンテナの各レコードがさらに次アドレス・フィールドを有し、それによりレコード・チェーンが、次アドレス・フィールドによって層 k データ・コンテナ内で定義され、シソーラス項目の少なくとも幾つかは、層 n データ・コンテナにあるレコード・チェーンにそれぞれ関連付けられ、それにより層 n の前記項目の1つに関するコード化データが、層 n データ・コンテナ内で自身に関連付けられたレコード・チェーンに記憶されるか、そこから検索可能である、請求項 36 から 43 のいずれか1項に記載の方法。

50

【請求項 4 5】

1 ≤ k < n の場合、前記シソーラス項目が、層 k データ・コンテナのレコード・チェーンとそれぞれ関連付けられ、それにより層 k の前記項目の 1 つに関するコード化データが、層 k データ・コンテナ内で自身に関連付けられたレコード・チェーンに記憶されるか、そこから検索可能である、請求項 4 4 に記載の方法。

【請求項 4 6】

1 < k ≤ n の場合、層 k データ・コンテナの各レコードが、さらに、層 k - 1 データ・コンテナにある個々のチェーンの第 1 のレコードのアドレスを指すために、頭部アドレス・フィールドを有する、請求項 4 4 に記載の方法。

【請求項 4 7】

1 ≤ k ≤ n の場合、各層 k データ・コンテナが、前記データ・コンテナのレコードの第 1 のフィールドおよび次アドレス・フィールドがそれぞれ記憶された少なくとも 2 つのファイルを備え、前記ファイルが別個にアクセス可能である、請求項 4 4 から 4 6 のいずれか 1 項に記載の方法。

【請求項 4 8】

データ・グラフに割り当てられた識別子をカバーする整数範囲が、複数の所定の部分に区分され、識別子リストを表すデータの少なくとも幾つかが、それぞれ前記部分に関連付けられた複数の記憶区間に分散され、前記部分の 1 つに関連付けられた記憶セクションが、前記部分の識別子で構成された識別子サブリストを表すデータを含む、先行する請求項のいずれか 1 項に記載の方法。

【請求項 4 9】

前記部分に関連付けられた記憶セクションを受け取るために、データ・グラフ識別子範囲の前記部分それぞれに、個々の記憶ユニットを設ける、請求項 4 8 に記載の方法。

【請求項 5 0】

ワード・シソーラスの少なくとも幾つかが、それぞれ前記部分に関連付けられた複数のセクションを有し、属性に関連付けられたワード・シソーラスの、前記部分の 1 つに関連付けられたセクションが、前記部分の識別子が割り当てられたデータ・グラフの前記属性に関連付けられたワードごとに個々の項目を有し、前記項目が、前記部分に関連付けられた記憶セクションから識別子サブリストを検索するためのデータを含む、請求項 4 9 に記載の方法。

【請求項 5 1】

マクロワード・シソーラスの少なくとも幾つかが、それぞれ前記部分に関連付けられた複数のセクションを有し、属性および接頭語長に関連付けられたマクロワード・シソーラスの、前記部分の 1 つに関連付けられたセクションが、前記接頭語長を有し、前記部分の識別子が割り当てられたデータ・グラフの前記属性に指定された少なくとも 1 つのワードの対応接頭語と一致する接頭語値ごとに個々の項目を有し、前記項目が、前記部分に関連付けられた記憶セクションから識別子サブリストを検索するためのデータを含む、請求項 1 2 および 4 9 の両方に記載の方法。

【請求項 5 2】

記憶セクションから個々の識別子サブリストを検索するデータを含むため、各シソーラス項目が、それぞれ前記部分に関連付けられた複数のフィールドを有する、請求項 4 8 または 4 9 に記載の方法。

【請求項 5 3】

データベース・システムで問合せを処理する方法で、属性のグループが定義され、データ集合体のワードが前記属性に指定され、属性のグループが、それぞれ独立した数のレコードを有する複数のデータ・テーブルに関連付けられた複数のサブグループに分割され、データ・テーブルからの個々のレコード間にリンクがあり、識別子がそれぞれデータ・グラフに割り当てられ、各データ・グラフが、それぞれ前記グループの属性に指定された関連属性値を表し、データ・グラフの各属性値が、デフォルト値または前記データ集合体のワードであり、それぞれが前記グループの個々の属性と関連付けられた複数のシソーラスと

10

20

30

40

50

、それぞれが前記シソーラスの項目と関連付けられたデータ・グラフ識別子リストを表すデータとが記憶され、1つの属性に関連付けられた各シソーラスが、前記1つの属性に指定できるワード・セットのサブセットへの区分に関して定義され、データ集合体の前記1つの属性に指定された少なくとも1つのワードを含むサブセットごとに個々の項目を有し、前記シソーラス項目に関連付けられたデータ・グラフ識別子リストが、前記1つの属性に指定された前記サブセットのワードを有するデータ・グラフに割り当てられた識別子を含み、

ー 問合せに関するシソーラス項目を含む組合せを決定するため、問合せ基準を分析するステップと、

ー 前記組合せに、および記憶され、前記関連シソーラス項目に関連付けられたデータ・グラフ識別子リストを表すデータに基づいて、マッチングデータ・グラフ識別子リストを決定するステップと、 10

ー 応答を出力するため、前記マッチングデータ・グラフ識別子リストを処理するステップとを含む方法。

【請求項54】

問合せ基準を分析するステップが、前記基準で言及された少なくとも1つの属性について、

ー 問合せ基準の前記属性について定義されたワードの少なくとも1つの範囲を選択することと、

ー データ集合体の前記属性に指定された選択範囲のワードを1つまたは複数のサブセットでマッピングすることとを含み、前記1つまたは複数のサブセットそれぞれのシソーラス項目が、選択範囲の関連項目として維持され、 20

マッチングデータ・グラフ識別子リストを決定するステップが、前記選択範囲について維持された関連シソーラス項目のデータによって表される識別子リストの個々の部分をマージすることを含む、請求項53に記載の方法。

【請求項55】

マッピングが、選択された範囲ごとに最小数の関連シソーラス項目を維持するように実行される、請求項54に記載の方法。

【請求項56】

属性に関連付けられた各シソーラスは、各サブセットが、前記属性に指定できるワードのセットの1ワードまたは連続ワードで構成されるように、区分に関して定義され、前記シソーラスの項目が、前記属性に指定されたワードに基づいてソートされ、問合せ基準を分析するステップが、関連シソーラス項目を識別するため、少なくとも1つのシソーラスに少なくとも1つの二分検索を含む、請求項54または55に記載の方法。 30

【請求項57】

シソーラスが、それぞれ1つのワードで構成されたサブセットへの区分に関して、それぞれグループの個々の属性に関連付けられたワード・シソーラスを備える、請求項56に記載の方法。

【請求項58】

デフォルト値が少なくとも1つのデータ・グラフに指定されたグループの属性に関連付けられた各ワード・シソーラスが、さらに、デフォルト値の項目を有し、それによって前記データ・グラフ識別子リストの1つが、デフォルト値に関して前記シソーラス項目に関連付けられ、前記属性に指定された前記デフォルト値を有するデータ・グラフに割り当てられた識別子を有する、請求項57に記載の方法。 40

【請求項59】

シソーラスが、さらに、前記グループの属性に、および前記属性のワード長より短い接頭語長に割り当てられた少なくとも1つのマクロワード・シソーラスを備え、それによって前記マクロワード・シソーラスが、前記接頭語長を有する共通接頭語で始まるワードでそれぞれ構成されたサブセットへの区分に関して定義される、請求項57または58に記載の方法。

【請求項 60】

前記グループの少なくとも1つの属性が、異なる接頭語長に割り当てられた複数のマクロワード・シソーラスを有する、請求項59に記載の方法。

【請求項 61】

問合せ基準を分析するステップが、関連シソーラス項目を含む前記組合せを、少なくとも1つの葉ノードを有する木として判断し、各葉ノードが、個々の属性に維持された少なくとも1つの関連シソーラス項目に対応する、請求項54から60のいずれか1項に記載の方法。

【請求項 62】

前記木が、前記少なくとも1つの葉ノードおよび少なくとも1つの演算子ノードを含む複数のノードを有し、各演算子ノードが、前記木の別のノードによって表された少なくとも1つの部分基準に適用されるブール演算子を表し、演算子ノードの1つが、問合せ基準を全て表す根ノードである、請求項61に記載の方法。 10

【請求項 63】

前記木のノードが、さらに、前記問合せ基準を分析するステップの前にデータ・グラフ識別子リストが決定されている少なくとも1つの事前設定ノードを含む、請求項62に記載の方法。

【請求項 64】

前記事前設定ノードのデータ・グラフ識別子リストが、以前の間合せの処理時に獲得された少なくとも1つのマッチングデータ・グラフ識別子リストから決定される、請求項63 20
に記載の方法。

【請求項 65】

マッチングデータ・グラフ識別子リストを決定するステップが、前記木の各ノードの個別識別子を獲得することを含み、これによって少なくとも1つの関連シソーラス項目に対応する各葉ノードについて獲得された識別子リストが、前記少なくとも1つの関連シソーラス項目に関連付けられた識別子リストの個々の部分のマージであり、少なくとも1つの部分的基準に適用されたブール演算子を表す各演算子ノードについて獲得された識別子リストが、前記少なくとも1つの部分的基準を表すノードについて獲得された識別子リストに前記ブール演算子を適用することによって獲得され、前記マッチングデータ・グラフ識別子リストが、根ノードについて獲得された識別子リストとして決定される、請求項62か 30
ら64のいずれか1項に記載の方法。

【請求項 66】

前記獲得された識別子リストがそれぞれ、前記データ・グラフに割り当てられた識別子が前記獲得されたリストに属するか示すために、個々のデータ・グラフに指定されたビットで構成されるビットマップ・ベクトルの形態で生成される、請求項65に記載の方法。

【請求項 67】

n個の連続コード化層を備えるコード化方式を使用して、シソーラス項目に関連付けられた識別子リストを表すコード化データを提供し、nが少なくとも1に等しい数であり、各層が、前記層の入力リストの整数を含む範囲をサブセットに分割するための所定のパターンを有し、前記識別子リストが、前記シソーラス項目の第1の層の入力リストであり、最終層以外の任意の層について、入力リストの少なくとも1つの整数を含む各サブセットの位置を、前記層のパターンで表す整数リストが、次層の入力リストを形成し、 40
コード化データが、層ごとに、および入力リストの少なくとも1つの整数を含むサブセットごとに、前記サブセットにおける入力リストの各整数の位置を表すデータを備え、少なくとも前記層が最終層である場合に、前記サブセットの位置を前記層のパターンで表すデータを備える、請求項65または66に記載の方法。

【請求項 68】

各層のパターンは、整数サブセットが同数の整数で構成された連続インターバルであるようなパターンである、請求項67に記載の方法。

【請求項 69】

前記整数の数が、層ごとに2の整数累乗である、請求項68に記載の方法。

【請求項70】

サブセットにおける入力リストの整数の位置を表す前記データがビットマップ・セグメントで構成される、請求項67から69のいずれか1項に記載の方法。

【請求項71】

マッチングデータ・グラフ識別子リストを決定するステップが、前記木のノードごとに層nの整数リストを決定することを含み、それによって葉ノードで決定された層nの整数リストが、前記葉ノードが対応する関連シソーラス項目で表された識別子リストのマージにコード化方式で関連付けられた層nの入力リストで構成され、それにより少なくとも1つの部分基準に適用されるブール演算子を表す演算子ノードごとに獲得される層nの整数リストが、前記ブール演算子を、前記少なくとも1つの部分基準を表すノードで決定された層nの整数リストに適用することによって獲得され、層nの結果リストが、根ノードについて獲得された層nの整数リストとして決定される、請求項67から70のいずれか1項に記載の方法。

【請求項72】

前記木のノードが、さらに、前記問合せ基準を分析するステップより前にデータ・グラフ識別子リストが決定されている少なくとも1つの事前設定ノードを含み、前記データ・グラフ識別子リストが、コード化方式に従い、前記事前設定ノードの前記層n整数リストとして決定された層n入力リストを提供する、請求項71に記載の方法。

【請求項73】

コード化方式で、コード化層nのサブセットにおける入力リストの各整数の位置を表すコード化データが、サブセットの個々の整数に関連付けられた各ビットを含む層nのビットマップ・セグメントを定義して、前記整数が前記入力リストに含まれるか示し、前記サブセットの位置を層nのパターンで表すデータは、前記層nビットマップ・セグメントに関連付けられた層n整数ランクを備え、葉ノードについて層n整数リストを決定するステップが、

- ー 層nビットマップ・ベクトルを論理ゼロで初期化することと、
- ー 前記葉ノードが対応する関連シソーラス項目ごとに、層nランクおよび関連するビットマップ・セグメントをコード化データから取得することと、
- ー 前記層nランクごとに、自身に関連する層nビットマップ・セグメントを、前記層nランクによって決定された位置を有する前記層nビットマップ・ベクトルのセグメントに重畳させることとを含み、重畳が、ビット毎のブールOR演算に従って実行され、葉ノードの前記層nリストが、結果の層nビットマップ・ベクトルに対応する、請求項71または72に記載の方法。

【請求項74】

$n > 1$ で、マッチングデータ・グラフ識別子リストを決定するステップが、さらに、 $n-1$ から1へと減少するkについて、前記木のノードごとに層kの整数リストを決定することを含み、それにより葉ノードについて決定された層k整数リストが、前記葉ノードが対応する関連シソーラス項目で表された識別子リストにコード化方式で関連付けられ、層k+1結果リストで表される位置を有する層kサブセットに属する層k入力リストの任意の整数で構成され、それによって少なくとも1つの部分基準に適用されるブール演算子を表す演算子ノードごとに獲得された層k整数リストが、前記ブール演算子を、前記少なくとも1つの部分基準を表すノードについて決定された層k整数リストに適用することによって獲得され、層k結果リストが、根ノードについて獲得された層k整数リストとして決定され、前記マッチングデータ・グラフ識別子リストが、決定された層1の結果リストに対応する、請求項71から73のいずれか1項に記載の方法。

【請求項75】

前記木のノードが、さらに、データ・グラフ識別子リストが前記問合せ基準を分析するステップの前に決定されている少なくとも1つの事前設定ノードを含み、前記データ・グラフ識別子リストが、コード化方式に従い、前記事前設定ノードについて前記層k整数リス

トとして決定される層 k 入力リストを提供する、請求項 74 に記載の方法。

【請求項 76】

コード化方式で、コード化層 $k < n$ の場合にサブセットにおける入力リストの各整数の位置を表すコード化データが、論理ゼロ前記整数が前記入力リストに属するか示すため、サブセットの個々の整数に関連付けられた各ビットを含む層 k ビットマップ・セグメントを定義し、コード化データが、さらに、前記層 k ビットマップ・セグメントに関連付けられた層 k 整数ランクを備えて、前記サブセットの位置を層 k パターンで表し、葉ノードについて層 k 整数リストを決定するステップが、

- 層 k ビットマップ・ベクトルを論理ゼロで初期化することと、
 - 前記葉ノードが対応する関連シソーラス項目ごとに、コード化データから層 k ランクを獲得することと、
 - 層 $k + 1$ 結果リストに属し、獲得された任意の層 k ランクを選択して、関連付けられた層 k ビットマップ・セグメントを、選択された層 k ランクによって決定された位置を有する前記層 k ビットマップ・ベクトルのセグメントに重畳することとを含み、重畳が、ビットごとのブール OR 演算に従って実行され、
- 葉ノードの前記層 k リストが、結果である層 k ビットマップ・ベクトルに対応する、請求項 74 または 75 に記載の方法。

【請求項 77】

$1 \leq k < n$ の場合、層 k ランク、および少なくとも 1 つのシソーラス項目について自身に関連付けられた層 k ビットマップ・セグメントが、個別の第 1 および第 2 のファイルの対応するアドレスに記憶され、葉ノードについて層 k 整数リストを決定するステップが、

- 前記第 1 および第 2 のファイルのアドレスに関連付けられたレコードを有するランク・テーブルを RAM メモリに設けることと、
- 前記第 1 のファイルで選択された層 k ランクのアドレスに関連付けられたランク・テーブル・レコードに、選択された層 k ランクを書き込むことにより、ランク・テーブルを充填することと、
- 層 k ランクを含み、第 2 のファイルのアドレスに関連付けられた充填ランク・テーブルの任意のレコードについて、第 2 のファイルの前記アドレスで関連する層 k ビットマップ・セグメントを読み出して、前記層 k ランクによって欠点された位置を有する前記層 k ビットマップ・ベクトルのセグメントに、読み出した層 k ビットマップ・セグメントを重畳させることとを含む、請求項 76 に記載の方法。

【請求項 78】

マッチングデータ・グラフ識別子リストを決定するステップが、さらに、 $1 < k \leq n$ になるような任意のコード化層 k について、コード化方式の層 k の入力リストとして結果リストの層 k を提供することにより獲得された層 k' 入力リストで構成された層 k' フィルタリング・リストを $k \leq k' \leq n$ について決定することを含み、

コード化方式で、コード化層 $k < n$ のサブセットにおける入力リストの各整数の位置を表すコード化データが、前記整数が前記入力リストに属するか示すために、サブセットの個々の整数に関連付けられた各ビットを含む層 k ビットマップ・セグメントを定義し、前記層 k ビットマップ・セグメントに関連付けられた層 k 整数ランクが、前記サブセットの位置を層 k パターンで表し、 $k < n$ の場合に、葉ノードについて層 k 整数リストを決定するステップが、

- ／ a ／ 層 k ビットマップ・ベクトルを論理ゼロで初期化することと、
- ／ b ／ 前記葉ノードが対応する関連シソーラス項目ごとに、コード化データから層 n ランクを獲得し、 $k' = n$ を設定することと、
- ／ c ／ 選択された層 k' ランクごとに、
- ／ c 1 ／ 選択された層 k' ランクが、層 k' フィルタリング・リストの少なくとも 1 つの整数を含むサブセットの位置を層 k' パターンで表す場合は、選択された層 k' ランクが関連付けられた層 k' ビットマップ・セグメントを獲得し、
- ／ c 2 ／ 前記層 k' ビットマップ・セグメントで表された位置を前記サブセット内に有

する層 k' フィルタリング・リストの整数がある場合は、その整数について、選択された層 k' ランク、および前記層 k' ビットマップ・セグメントで表された前記位置から決定した個々の層 $k' - 1$ ランクを選択し、

／ c 3 ／ $k' > k + 1$ の場合は、 k' を 1 ユニットだけ減少させてステップ／ c ／を実行し、

／ c 4 ／ $k' - 1 = k$ の場合は、選択された層 $k' - 1$ ランクが関連付けられた層 k ビットマップ・セグメントを獲得し、前記選択層 $k' - 1$ ランクによって決定された位置を有する前記層 k ビットマップ・ベクトルのセグメントに前記層 k ビットマップ・セグメントを重畳することを含み、重畳が、ビットごとのブール OR 演算に従って実行され、葉ノードの前記層 k リストが、結果である層 k ビットマップ・ベクトルに対応する、請求項 74 または 75 に記載の方法。

10

【請求項 79】

$1 \leq k < n$ の場合、少なくとも 1 つのシソーラス項目に関する層 k ビットマップ・セグメントが、それぞれ自身に関連付けられた層 k ランクに対応するアドレスで少なくとも 1 つの層 k ファイルに記憶され、 $1 \leq k < n$ の場合に、葉ノードについて層 k 整数リストを決定するステップが、

ー 前記層 k ファイルのアドレスに関連付けられたレコードを有するランク・テーブルを RAM メモリに設けることと、

ー 選択された層 k ランクに対応するアドレスに関連付けられたランク・テーブル・レコードに、選択された層 k ランクを書き込むことにより、ランク・テーブルを充填することと、

20

ー 層 k ランクを含み、前記層 k ファイルのアドレスに関連付けられた充填ランク・テーブルの任意のレコードについて、前記アドレスにて関連する層 k ビットマップ・セグメントを読み出して、前記層 k ランクによって決定された位置を有する前記層 k ビットマップ・ベクトルのセグメントに、読み出した層 k ビットマップ・セグメントを重畳させることを含む、請求項 78 に記載の方法。

【請求項 80】

それぞれがデータ・グラフに関連付けられた複数の行、およびそれぞれが属性サブグループに関連する複数の列とを有するリンク・テーブルが記憶され、リンク・テーブルの各行が、列ごとに、前記行に関連付けられたデータ・グラフで表され、前記列の 1 つに関連するサブグループの属性に指定された各属性値がデフォルト値であることを示す値か、または前記行に関連付けられたデータ・グラフで表され、前記列の 1 つに関連するサブグループの属性に指定されたデータ・グラフで表されたデータ集合体から記憶された少なくとも 1 つのワードを検索するためのリンク値を含み、マッチングデータ・グラフ識別子リストを処理するステップが、マッチングデータ・グラフ識別子リストで識別されたデータ・グラフに関連付けられたリンク・テーブルの任意の行で、少なくとも 1 つの値を読み出すことを含む、請求項 53 から 79 のいずれか 1 項に記載の方法。

30

【請求項 81】

前記データ・テーブルが記憶され、属性サブグループに関連付けられたリンク・テーブルの列に含まれる各リンク値が、前記サブグループに関連付けられたデータ・テーブルのレコードを識別するためのデータを備え、マッチングデータ・グラフ識別子リストを処理するステップが、リンク・テーブルの行で読み出されたリンク値によって識別された任意のデータ・テーブル・レコードの少なくとも一部を読み出すことを含む、請求項 80 に記載の方法。

40

【請求項 82】

マッチングデータ・グラフ識別子リストを処理するステップが、問合せで指定された少なくとも 1 つの属性について、前記属性に関連付けられたシソーラスを選択することと、マッチングデータ・グラフ識別子リストとの非空白交差を有する識別子リストが関連付けられた選択シソーラスの項目を検出することを含む、請求項 53 から 79 のいずれか 1 項に記載の方法。

50

【請求項 8 3】

問合せで指定された前記属性が、異なる接頭語長に関連付けられ、記憶された $Q + 1$ 個のシソーラスを有し、 Q は少なくとも 0 に等しい整数であり、前記 $Q + 1$ 個のシソーラスがそれぞれ、 $0 \leq q \leq Q$ になるようなシソーラス・レベル・パラメータ q を有し、それにより接頭語長が、レベル・パラメータ q の減少関数であって、 $q = 0$ の場合は前記属性のワード長に対応し、前記 $Q + 1$ 個のシソーラスがそれぞれ、前記シソーラスに関連付けられた接頭語長を有する共通の接頭語で開始するワードでそれぞれ構成されたサブセットへの個々の区分に関して定義され、前記シソーラスの項目が、接頭語長に基づき記憶される、請求項 8 2 に記載の方法。

【請求項 8 4】

10

レベル・パラメータ $Q A \geq 0$ を有する選択されたシソーラスで、選択されたシソーラスの項目の検出が、

／ a ／ $Q A \leq q \leq Q$ の場合に、個々のレベル q 目標リスト、およびレベル q シソーラスの連続項目を含む個々のレベル q シソーラス範囲を設けるステップと、

／ b ／ レベル Q 目標リストをマッチングデータ・グラフ識別子リストで初期化し、レベル・パラメータ q を値 Q で初期化して、レベル Q シソーラス範囲の第 1 の項目を選択するステップと、

／ c ／ レベル q 目標リストとレベル q シソーラス範囲の選択項目に関連付けられた識別子リストとの間の交差リストを決定するステップと、

／ d ／ 先行ステップ／ c ／で決定した交差リストが空白である場合に、レベル q シソーラス範囲の別の項目を選択して、ステップ／ c ／を繰り返すステップと、 20

／ e ／ q が $Q A$ より大きい場合、

／ e 1 ／ レベル $q - 1$ 目標リストを、先行ステップ／ c ／で決定した交差リストと等しく設定し、

／ e 2 ／ レベル $q - 1$ シソーラス範囲を、選択されたレベル q シソーラス項目のレベル q 接頭語で開始するレベル $q - 1$ 接頭語に関するレベル $q - 1$ シソーラスの項目で構成するものと設定して、レベル $q - 1$ シソーラス範囲の第 1 の項目を選択し、

／ e 3 ／ q を 1 ユニットだけ減少させて、ステップ／ c ／に戻るステップと、

／ f ／ q が $Q A$ と等しい場合、

／ f 1 ／ 検出した項目に選択したレベル $Q A$ シソーラス項目を含み、 30

／ f 2 ／ レベル Q 目標リストが、先行ステップ／ c ／で決定された交差リストと等しい場合は、選択されたシソーラスでの項目の検出を終了し、

／ f 3 ／ 前記交差リストにない少なくとも 1 つの整数を含む任意の目標リストから、先行ステップ／ c ／で決定された交差リストの整数を除去し、

／ f 4 ／ q を、目標リストが前記交差リストにない少なくとも 1 つの整数を含む最小レベル・パラメータと設定し、

／ f 5 ／ レベル q シソーラス範囲の別の項目を選択して、ステップ／ c ／に戻るステップとを含む、請求項 8 3 に記載の方法。

【請求項 8 5】

$Q \geq 1$ で、レベル・パラメータ $q \geq 1$ を有する各シソーラスが、さらに、レベル q 接頭語値に与えられた項目ごとに、前記レベル q シソーラス項目のレベル q 接頭語で開始する最低または最高レベル $q - 1$ 接頭語に対応するレベル $q - 1$ シソーラスの項目を指定するデータを含み、ステップ／ e 2 ／が、選択されたレベル q シソーラス項目に指定されたレベル $q - 1$ シソーラス項目を選択することを含む、請求項 8 4 に記載の方法。 40

【請求項 8 6】

n 個の連続コード化層を含むコード化方式を使用して、 $0 \leq q \leq Q$ の場合に、レベル q シソーラス項目に関連付けられた識別子リストを表すコード化データを提供し、 n が少なくとも 1 に等しい数字であり、各層が、前記層の入力リストの整数を含む範囲をサブセットに分割するための所定のパターンを有し、前記識別子リストが、前記シソーラス項目の第 1 の層の入力リストであり、最終層以外の任意の層について、入力リストの少なくとも 1 50

つの整数を含む各サブセットの位置を前記層のパターンで表す整数リストが、次層の入力リストを形成して、

コード化データが、層ごとに、および入力リストの少なくとも1つの整数を含むサブセットごとに、前記サブセットにおける入力リストの各整数の位置を表すデータを備え、少なくとも前記層が最終層である場合には、前記サブセットの位置を前記層のパターンで表すデータを備え、

各レベル q 目標リストが層1およびレベル q フィルタリング・リストを形成し、 $Q \leq q \leq Q$ の場合に、コード化方式の層1入力リストとして提出されて、 $n > 1$ の場合は、 $1 < k \leq n$ について個々の層 k およびレベル q フィルタリング・リストを提供し、レベル q 目標リストから提供された前記層 k およびレベル q フィルタリング・リストが、コード化方式で前記レベル q 目標リストから獲得された層 k 入力リストである、請求項84または85に記載の方法。

10

【請求項87】

各層のパターンは、整数サブセットが、同数の整数で構成された連続インターバルであるようなパターンである、請求項86に記載の方法。

【請求項88】

前記整数の数が、各層の2の整数累乗である、請求項87に記載の方法。

【請求項89】

レベル q 目標リストと識別子リストとの間の交差リストを決定するステップ／c／が、 $k = n$ から、

20

／c1／ コード化方式で前記識別子リストから獲得された層 k 入力リストと、前記レベル q 目標リストに対応する層 k およびレベル q フィルタリング・リストとの間の層 k 交差リストを計算することと、

／c2／ 計算した層 k 交差リストが空白である場合は、レベル q 目標リストと識別子リストとの間の前記交差リストを空白であると判断することと、

／c3／ $k = 1$ の場合は、レベル q 目標リストと識別子リストとの間の前記交差リストを、計算された層1交差リストであると判断することと、

／c4／ $k > 1$ の場合は、 k を1ユニット減少させて、ステップ／c1／から繰り返すこととを含む、請求項86から88のいずれか1項に記載の方法。

【請求項90】

30

コード化方式で、コード化層 $k \leq n$ のサブセットにおける入力リストの各整数の位置を表すコード化データが、論理ゼロ前記整数が前記入力リストに属するか示すため、サブセットの個々の整数に関連付けられた各ビットを含む層 k ビットマップ・セグメントを定義し、前記サブセットの位置を層 k パターンで表すデータが、前記層 k ビットマップ・セグメントに関連付けられた層 k 整数ランクを備え、コード化方式で識別子リストから獲得された層 k 入力リストと、第1の層 k ビットマップ・ベクトルによって表された層 k とレベル q フィルタリング・リストとの間の層 k 交差リストを計算するステップ／c1／が、

— 第2の層 k ビットマップ・ベクトルを論理ゼロで初期化することと、

— 前記識別子リストを表すコード化データから層 k ランクを獲得することと、

— 獲得された中から、前記層 k およびレベル q のフィルタリング・リストの少なくとも1つの整数を含むサブセットの位置を層 k パターンで表す任意の層 k ランクを選択して、選択された層 k ランクに関連付けられた層 k ビットマップ・セグメントを獲得し、獲得された層 k ビットマップ・セグメントを、ビットごとのブールAND演算に従って選択された層 k ランクで決定された位置を有する第1の層 k ビットマップ・ベクトルのセグメントと組み合わせることにより、選択された層 k ランクによって決定された位置を有する第2の層 k ビットマップ・ベクトルのセグメントを決定することとを含む、

40

前記層 k 交差リストが、結果である第2の層 k ビットマップ・ベクトルに対応する、請求項89に記載の方法。

【請求項91】

$1 \leq k < n$ の場合に、層 k ランク、および少なくとも1つのシソーラス項目について自身

50

に関連付けられた層 k ビットマップ・セグメントが、別個の第 1 および第 2 のファイルの対応するアドレスに記憶され、コード化方式で識別子リストから獲得された層 k 入力リストと、第 1 の層 k ビットマップ・ベクトルで表された層 k およびレベル q のフィルタリング・リストとの間の層 k 交差リストを計算するステップ／c 1／が、

— 前記第 1 および第 2 のファイルのアドレスに関連付けられたレコードを有するランク・テーブルを RAM メモリに設けることと、

— 前記第 1 のファイルの前記選択層 k ランクのアドレスに関連付けられたランク・テーブル・レコードに、選択された層 k ランクを書き込むことにより、ランク・テーブルを充填することと、

— 層 k ランクを含み、第 2 のファイルのアドレスに関連付けられた充填ランク・テーブルの任意のレコードについて、第 2 のファイルの前記アドレスにて関連する層 k ビットマップ・セグメントを読み出して、読み出した層 k ビットマップ・セグメントを、ビットごとのブール AND 演算に従って前記層 k ランクによって決定された位置を有する第 1 の層 k ビットマップ・ベクトルのセグメントと組み合わせて、前記層 k ランクによって決定された位置を有する第 2 の層 k ビットマップ・ベクトルのセグメントを決定することを含む、請求項 90 に記載の方法。 10

【請求項 92】

$n > 1$ の場合、コード化方式にて、コード化層 $k \leq n$ についてサブセットにおける入力リストの各整数の位置を表すコード化データが、前記整数が前記入力リストに属するか示すために、サブセットの個々の整数と関連付けられた各ビットを含む層 k ビットマップ・セグメントを定義し、前記層 k ビットマップ・セグメントに関連付けられた層 k 整数ランクが、前記サブセットの位置を層 k パターンで表し、 $1 \leq k \leq n$ の場合に個々の第 1 の層 k ビットマップ・ベクトルによって表された層 k およびレベル q のフィルタリング・リストに対応するレベル q 目標リストと識別子リストとの間の交差リストを決定するステップ／c 1／が、 20

／c 1／ 第 2 のビットマップ・ベクトルを論理ゼロで初期化することと、

／c 2／ 前記識別子リストを表すコード化データから獲得された層 n ランクを選択して、 $k = n$ を設定することと、

／c 3／ 選択された層 k ランクごとに、

／c 3 1／ 選択された層 k ランクが、前記層 k およびレベル q のフィルタリング・リストの少なくとも 1 つの整数を含むサブセットの位置を層 k パターンで表す場合は、選択された層 k ランクが関連付けられた層 k ビットマップ・セグメントを獲得し、 30

／c 3 2／ 前記層 k ビットマップ・セグメントで表された位置を前記サブセット内に有する層 k およびレベル q のフィルタリング・リストの整数がある場合は、その整数について、選択された層 k ランク、および前記層 k ビットマップ・セグメントで表された前記位置から決定した個々の層 $k - 1$ ランクを選択し、

／c 3 3／ $k > 2$ の場合は、 k を 1 ユニットだけ減少させてステップ／c 3／を実行し、

／c 3 4／ $k = 2$ の場合は、選択された層 1 ランクが関連付けられた層 1 ビットマップ・セグメントを獲得し、獲得した層 1 ビットマップ・セグメントを、ビットごとのブール AND 演算に従って前記層 1 ランクで決定した位置を有する第 1 の層 1 ビットマップ・ベクトルのセグメントと組み合わせて、前記層 1 ランクによって決定された位置を有する第 2 のビットマップ・ベクトルのセグメントを決定することを含み、 40

前記交差リストが、結果の第 2 のビットマップ・ベクトルに対応する、請求項 86 から 88 のいずれか 1 項に記載の方法。

【請求項 93】

少なくとも 1 つのシソーラス項目の層 1 ビットマップ・セグメントが、自身に関連付けられた層 1 ランクにそれぞれ対応するアドレスで、少なくとも 1 つの層 1 ファイルに記憶され、交差リストを決定するステップ／c 1／が、

— 前記層 1 ファイルのアドレスに関連付けられたレコードを有するランク・テーブルを 50

R A Mメモリに設けることと、

ー 選択された層 1 ランクに対応するアドレスに関連付けられたランク・テーブル・レコードに、ステップ／c 3 2／で選択された層 1 ランクを書き込むことによりランク・テーブルを充填することと、

ー 層 1 ランクを含み、前記層 1 ファイルのアドレスに関連付けられた充填ランク・テーブルの任意のレコードについて、前記アドレスにて関連する層 1 ビットマップ・セグメントを読み出して、読み出した層 1 ビットマップ・セグメントを、ビットごとのブール A N D 演算に従って前記層 1 ランクによって決定された位置を有する第 1 の層 1 ビットマップ・ベクトルのセグメントと組み合わせて、前記層 1 ランクによって決定された位置を有する第 2 の層 1 ビットマップ・ベクトルのセグメントを決定することを含む、請求項 9 2 に記載の方法。

【請求項 9 4】

さらに、レベル q シソーラスの項目ごとに事前フィルタリング・フラグを決定することを含み、前記項目が、層 n およびレベル q のフィルタリング・リストに少なくとも 1 つの整数を含むサブセットの位置を層 n パターンで表す層 n ランクを一切定義していないコード化データによって表されたデータ・グラフ識別子リストに関連付けられている場合、前記事前フィルタリング・フラグは第 1 の値を有し、前記層 n およびレベル q のフィルタリング・リストに対応するレベル q の目標リストと、レベル q シソーラスの項目に関連付けられた識別子リストとの間の交差リストを決定するステップ／c／が、前記項目について決定された事前フィルタリング・フラグが前記第 1 の値を有する場合に、前記交差リストを空白であると決定することを含む、請求項 9 0 から 9 3 のいずれか 1 項に記載の方法。

【請求項 9 5】

層 n およびランク q のフィルタリング・リストに少なくとも 1 つの整数を含むサブセットの位置を層 n パターンで表す層 n ランクを定義するコード化データによって表されたデータ・グラフ識別子リストに関連付けられたレベル q シソーラスがある場合、その項目について、前記層 n ランクに関連付けられた層 n ビットマップ・セグメントが獲得され、獲得された層 n ビットマップ・セグメントが、前記サブセット内で、前記層 n およびレベル q のフィルタリング・リストの整数の位置を一切表さない場合は、前記第 1 の値が、前記項目について決定された事前フィルタリング・フラグに割り当てられる、請求項 9 4 に記載の方法。

【請求項 9 6】

マッチングデータ・グラフ識別子リストを処理するステップが、さらに、選択されたシソーラスに検出項目があれば、それに関連付けられた出力データを出力テーブルに書き込むことを含む、請求項 8 2 から 9 5 のいずれか 1 項に記載の方法。

【請求項 9 7】

出力テーブルが、マッチングデータ・グラフ識別子リストの各識別子に対応する個々の行を含み、選択シソーラスの検出された項目に関連付けられた出力データが、マッチングデータ・グラフ識別子リストと、前記検出シソーラス項目に関連付けられた識別子リストとの両方に属するデータ・グラフ識別子に対応する出力テーブルの任意の行に書き込まれる、請求項 9 6 に記載の方法。

【請求項 9 8】

各データ・グラフ識別子が、自身に対応する出力テーブルの個々の行を有し、出力テーブルの行が、出力データを書き込む前にデフォルト値で初期化され、デフォルト値を含まない出力テーブルの行が、出力データの書き込み後に読み出される、請求項 9 7 に記載の方法。

【請求項 9 9】

出力テーブルが、データ・グラフ識別子ごとに個々のレコードを有する索引ファイルに関連付けられ、デフォルト値、または前記データ・グラフ識別子に対応する出力テーブルの個々の行を指定するポインタを含み、索引ファイルのレコードが、出力データの書き込み前にデフォルト値で初期化され、マッチングデータ・グラフ識別子リストと、第 1 の選択

10

20

30

40

50

シソーラスの前記検出項目に関連付けられた識別子リストとの両方に属するデータ・グラフ識別子ごとに、第1の選択シソーラスの検出項目に関連付けられた出力データを書き込むステップが、

- ー 前記データ・グラフ識別子に対応して、出力テーブルの使用可能な行を割り当てることと、
- ー 出力データを割り当てた行に書き込むことと、
- ー 割り当てた行を指すポインタを、前記データ・グラフ識別子のために設けた索引ファイルのレコードに書き込むこととを含む、請求項97に記載の方法。

【請求項100】

出力テーブルが、シソーラスが選択された個々の属性にそれぞれ関連付けられた複数の列を有し、出力テーブルの列に関連付けられた属性について選択されたシソーラスの検出項目に関連付けられた出力データが、前記列に書き込まれる、請求項97から99のいずれか1項に記載の方法。 10

【請求項101】

少なくとも1つの第2の選択シソーラスの検出項目に関連付けられた出力データを書き込むステップが、マッチングデータ・グラフ識別子リストと、第2の選択シソーラスの前記検出項目に関連付けられた識別子リストとの両方に属するデータ・グラフ識別子ごとに、

- ー 前記データ・グラフ識別子のために設けた索引ファイルのレコードに含まれたポインタを読み出すことと、
- ー 出力データを、前記ポインタによって指定された出力テーブルの行に書き込むこととを含む、請求項99および100の両方に記載の方法。 20

【請求項102】

選択されたシソーラスが、それぞれ1ワードで構成されたサブセットへの区分に関して定義されたワード・シソーラスであり、検出項目に関連付けられた出力データがワードを備え、そのワードのために前記検出項目が設けられている、請求項96から101のいずれか1項に記載の方法。

【請求項103】

選択されたシソーラスが、接頭語長に関連付けられ、それぞれが前記接頭語長を有する共通接頭語で始まるワードで構成されたサブセットへの区分に関して定義されたマクロワード・シソーラスであり、検出項目に関連付けられた出力データが接頭語値を備え、その接頭語値のために前記検出項目が設けられている、請求項96から101のいずれか1項に記載の方法。 30

【請求項104】

検出項目に関連付けられた出力データが、選択シソーラスの前記検出項目のアドレスを備える、請求項96から101のいずれか1項に記載の方法。

【請求項105】

選択シソーラスの検出項目に関連付けられた出力データが、前記シソーラス項目から導出された数値を備える、請求項96から101のいずれか1項に記載の方法。

【請求項106】

少なくとも1つの選択シソーラスの検出項目について、前記数値が、前記項目に記憶したシソーラス値に数学的関数を適用することによって計算される、請求項105に記載の方法。 40

【請求項107】

少なくとも1つの選択シソーラスの検出項目について、前記数値が、前記項目に記憶されたシソーラス値、および出力テーブルに既に存在する少なくとも1つの値を含む複数の値に、数学的関数を適用することによって計算される、請求項105または106に記載の方法。

【請求項108】

出力テーブルが、マッチングデータ・グラフ識別子リストの各識別子に対応する個々の行を含み、検出されたシソーラス項目から導出された数値が、マッチングデータ・グラフ識 50

別子リストと、前記検出シソーラス項目に関連付けられた識別子リストとの両方に属するデータ・グラフ識別子に対応する出力テーブルの任意の行に書き込まれる、請求項105から107のいずれか1項に記載の方法。

【請求項109】

第1の選択シソーラスの検出項目から導出され、マッチングデータ・グラフ識別子リストと、第1の選択シソーラスの前記項目に関連付けられた識別子リストとの両方に属するデータ・グラフ識別子に対応する出力テーブルの任意の行に書き込まれた数値が、前記項目に記憶されたシソーラス値から獲得され、

少なくとも1つの第2の選択シソーラスの検出項目から導出され、マッチングデータ・グラフ識別子リストと、第2の選択シソーラスの前記項目に関連付けられた識別子リストとの両方に属するデータ・グラフ識別子に対応する出力テーブルの行に書き込まれた数値が、前記項目に記憶されたシソーラス値、および出力テーブルの前記行に既に存在する少なくとも1つの値を含む複数の値に数学的関数を適用することによって計算される、請求項108に記載の方法。

【請求項110】

さらに、それぞれ出力テーブルの行に書き込まれた数値のセットから出力値を計算することを含む、請求項108または109に記載の方法。

【請求項111】

データ・グラフに割り当てられた識別子を含む整数範囲が、複数の所定の部分に区分され、識別子リストを表すデータの少なくとも一部が、それぞれ前記部分に関連付けられた複数の記憶セクションに分散し、前記部分の1つに関連付けられた記憶セクションが、前記部分の識別子で構成された識別子サブリストを表すデータを含み、

マッチングデータ・グラフ識別子リストを決定するステップが、個々の記憶セクションによって、データ・グラフ識別子範囲の異なる部分ごとに別個に実行される、請求項53から110のいずれか1項に記載の方法。

【請求項112】

マッチングデータ・グラフ識別子処理するステップが、個々の記憶セクションによって、データ・グラフ識別子範囲の異なる部分ごとに、少なくとも一部は別個に実行される、請求項111に記載の方法。

【請求項113】

シソーラスが、それぞれ前記部分に関連付けられた複数のセクションを有し、属性に関連付けられ、サブセットへの区分に関して定義されたシソーラスの、前記部分の1つに関連付けられたセクションが、前記区分のサブセットごとに、前記部分の識別子が割り当てられたデータ・グラフの前記属性に指定された少なくとも1つのワードを含む個々の項目を有し、前記項目が、前記属性に指定された前記サブセットのワードを有するデータ・グラフに割り当てられた前記部分の各識別子を含む識別子サブリストを表すデータを含み、問合せ基準を分析するステップが、個々のシソーラス・セクションによって、データ・グラフ識別子範囲の異なる部分ごとに、少なくとも一部は別個に実行される、請求項111または112に記載の方法。

【請求項114】

別個のステップ実行が、データ・グラフ識別子範囲の異なる部分ごとに、個々のプロセッサによって並行して実行される、請求項111から113のいずれか1項に記載の方法。

【請求項115】

各シソーラス項目が、記憶セクションから個々の識別子サブリストを検索するためのデータを含むため、前記部分に個々に関連付けられた複数のフィールドを有し、問合せ基準を分析するステップが、データ・グラフ識別子範囲の全ての部分について中央で実行され、記憶セクションによってマッチングデータ・グラフ識別子リストを決定するステップを実行するプロセッサが使用する関連シソーラス項目が、前記記憶セクションから識別子サブリストを検索するため、データによって指定される、請求項114に記載の方法。

【請求項116】

問合せ基準を分析するステップが、通信網を通して前記プロセッサに接続された問合せサーバによって実行される、請求項 1 1 5 に記載の方法。

【請求項 1 1 7】

リスト更新サーバが、通信網を通して、個々に前記プロセッサに結合された複数の記憶ユニットに接続され、リスト更新サーバが記憶ユニットを制御して、記憶セクションを維持する、請求項 1 1 4 から 1 1 6 のいずれか 1 項に記載の方法。

【請求項 1 1 8】

データベース・システムで情報を編成する方法で、個々にデータ属性に関連付けられた列を有する参照テーブルの個々の行を指定するため、複数の行識別子が定義され、前記行が、データ集合体にある前記属性に指定された関連ワードのグループを含み、

10

— 属性の 1 つと、および前記属性のワード長より短い接頭語長とに関連付けられた少なくとも 1 つのマクロワード・シソーラスを記憶するステップを含み、前記マクロワード・シソーラスが、前記接頭語長を有し、データ集合体にある前記データ属性に指定された少なくとも 1 つのワードの対応する接頭語と一致する接頭語値ごとに個々の項目を有し、さらに、

— 個々にマクロワード・シソーラス項目に関連付けられた識別子リストを表すデータを記憶するステップを含み、属性に関連付けられたマクロワード・シソーラスの、接頭語値に関する項目に関連付けられた識別子リストが、前記属性に関連付けられた列に、前記接頭語値と一致する対応接頭語を含むワードを有する参照テーブルの行を指定する行識別子を含む方法。

20

【請求項 1 1 9】

属性に関連付けられた各マクロワード・シソーラスの項目が、接頭語値に基づいてソートされる、請求項 1 1 8 に記載の方法。

【請求項 1 2 0】

異なる接頭語長に関連付けられた複数のマクロワード・シソーラスが、少なくとも 1 つの属性のために記憶される、請求項 1 1 8 または 1 1 9 に記載の方法。

【請求項 1 2 1】

さらに、前記属性の 1 つに関連付けられたワード・シソーラスを記憶するステップを含み、前記ワード・シソーラスが、データ集合体に、前記属性に少なくとも 1 回指定されたワードごとに個々の項目を有し、前記項目が、前記属性に関連付けられた列に、前記ワードを有する参照テーブルの行を指定する各行識別子を含む識別子リストを表すデータを含む、請求項 1 1 8 から 1 2 0 のいずれか 1 項に記載の方法。

30

【請求項 1 2 2】

参照テーブルが少なくとも 1 つの行にデフォルト値を有するための属性に関連付けられたワード・シソーラスが、デフォルト値の項目を有し、前記属性に関連付けられた列に、前記デフォルト値を有する参照テーブルの行を指定する各行識別子を含む識別子リストを表すデータを含む、請求項 1 2 1 に記載の方法。

【請求項 1 2 3】

ワード・シソーラスの項目が、前記属性に指定されたワードに基づいてソートされる、請求項 1 2 1 または 1 2 2 に記載の方法。

40

【請求項 1 2 4】

少なくとも 1 つの属性が、異なる接頭語長に割り当てられ、記憶された Q 個のマクロワード・シソーラスを有し、それぞれが、 $1 \leq q \leq Q$ になるよう、シソーラス・レベル・パラメータ q を有し、 Q は少なくとも 1 に等しい整数であり、接頭語長が、 $Q > 1$ の場合のレベル・パラメータの減少関数であり、レベル 1 のマクロワード・シソーラスが、さらに、レベル 1 接頭語値のために設けた項目ごとに、前記レベル 1 接頭語値と一致する対応接頭語を含む最低または最高ワードに対応する前記属性に関連付けられたワード・シソーラスの項目を指定するデータを含み、レベル・パラメータ $q > 1$ を有する任意のマクロワード・シソーラスが、さらに、レベル q 接頭語値のために設けた項目ごとに、前記レベル q 接頭語値と一致する対応接頭語を含む最低または最高レベル $q - 1$ マクロワードに対応する

50

レベル $q-1$ マクロワード・シソーラスの項目を指定するデータを含む、請求項 119 および 123 の両方に記載の方法。

【請求項 125】

前記参照テーブルが、記憶されていない仮想テーブルである、請求項 118 から 124 のいずれか 1 項に記載の方法。

【請求項 126】

さらに、それぞれが参照テーブルの行に関連付けられた複数の行と、それぞれが属性サブグループに関連する複数の列とを有するリンク・テーブルを記憶するステップを含み、リンク・テーブルの各行が、1つの列ごとに、関連する参照テーブルで表示され、前記 1つの列に関連付けられたサブグループの属性に指定された各属性がデフォルト値であることを示す値か、または関連付けられた参照テーブル行に表示され、前記 1つの列に関連付けられたサブグループの属性に指定されたデータ集合体の記憶ワードから少なくとも 1つを検索するためのリンク値を含む、請求項 125 に記載の方法。 10

【請求項 127】

個々のデータ・テーブルが、属性サブグループごとに記憶され、属性サブグループに関連付けられたリンク・テーブルの列に含まれる各リンク値が、前記サブグループのために記憶されたデータ・テーブルの行を識別するデータを備える、請求項 126 に記載の方法。

【請求項 128】

識別子リストを表す前記データが、少なくとも 1つのシソーラス項目について、 n 個の連続するコード化層を有するコード化方式によって獲得されたコード化データを備え、 n が少なくとも 1 に等しい数字であり、各層が、前記層の入力リストの整数を含む範囲をサブセットに分割する所定のパターンを有し、前記識別子リストが、前記シソーラス項目の第 1 の層の入力リストであり、最終層以外の任意の層について、入力リストの少なくとも 1つの整数を含む各サブセットの位置を前記層のパターンで表す整数リストが、次層の入力リストを形成し、前記コード化データが、層ごとに、および入力リストの少なくとも 1つの整数を含むサブセットごとに、前記サブセットにおける入力リストの各整数の位置を表すデータを備え、さらに少なくとも前記層が最終層の場合は、前記サブセットの位置を前記層のパターンで表すデータを備える、請求項 118 から 127 のいずれか 1 項に記載の方法。 20

【請求項 129】

コード化データが、それぞれシソーラス項目に割り当てられたファイルを含む複数のファイルに記憶される、請求項 128 に記載の方法。 30

【請求項 130】

コード化データが、前記シソーラスの項目に関連するコード化データを含むために、個々のシソーラスに割り当てられた少なくとも 1つのファイルを含む複数のファイルに記憶される、請求項 128 または 129 に記載の方法。

【請求項 131】

コード化データが、前記複数のシソーラスの項目に関連するコード化データを含むため、複数のシソーラスに割り当てられた少なくとも 1つのファイルに記憶される、請求項 128 から 130 のいずれか 1 項に記載の方法。 40

【請求項 132】

各層のパターンは、整数サブセットが、同数の整数で構成された連続的インターバルであるようなパターンである、請求項 128 から 131 のいずれか 1 項に記載の方法。

【請求項 133】

前記整数の数が、層ごとに 2 の整数累乗である、請求項 132 に記載の方法。

【請求項 134】

$n \geq 2$ で、それぞれが複数のレコードを有する層 k のデータ・コンテナが、 $1 \leq k \leq n$ の場合にコンピュータ・メモリに設けられ、層 k データ・コンテナの各レコードが、サブセットの位置を層 k パターンで表す層 k の整数ランクに関連付けられ、サブセットの位置を層 k パターンで表す層 k ランクに関連付けられた層 k データ・コンテナの各レコードが、 50

識別子リストに関連する層 k 入力リストの任意の整数の前記サブセットにおける位置を検索するためのデータを含む第 1 のフィールドを有し、前記層 k ランクと、前記第 1 のフィールドに含まれるデータから検索可能な任意の位置との組合せが、 $k > 1$ の場合に、層 $k - 1$ データ・コンテナの個々のレコードが関連付けられた層 $k - 1$ ランクを決定し、 $k = 1$ の場合に、前記識別子リストの識別子を決定する、請求項 1 2 8 から 1 3 3 のいずれか 1 項に記載の方法。

【請求項 1 3 5】

$1 \leq k \leq n$ の場合、サブセットにおける層 k 入力リストの任意の整数の位置を検索するために層 k データ・コンテナのレコードの第 1 のフィールドに含まれる前記データが、前記整数が前記層 k 入力リストに属するか示すため、前記サブセットの個々の整数に関連付けられた各ビットを含むビットマップ・セグメントを備える、請求項 1 3 4 に記載の方法。 10

【請求項 1 3 6】

$1 \leq k \leq n$ の場合、層 k ランクに関連付けられた層 k データ・コンテナの各レコードが、さらに、前記層 k ランクを含むための第 2 のフィールドを有する、請求項 1 3 5 に記載の方法。

【請求項 1 3 7】

各データ・コンテナが、前記データ・コンテナのレコードの第 1 および第 2 のフィールドがそれぞれ記憶された少なくとも 2 つのファイルを備え、前記ファイルが別個にアクセス可能である、請求項 1 3 6 に記載の方法。

【請求項 1 3 8】 20

$1 \leq k < n$ の場合は、層 k データ・コンテナの各レコードが、さらに、層 $k + 1$ パターンのサブセットにおける層 $k + 1$ 入力リストの整数の位置を表す数字を含むための第 2 のフィールドを有し、

$1 < k \leq n$ の場合は、層 k パターンのサブセットにおける層 k 入力リストの任意の整数の位置を検索するために、層 k ランクに関連付けられた層 k データ・コンテナのレコードの第 1 のフィールドに含まれた前記データが、層 k パターンの前記サブセットにおける前記層 k 入力リストの整数の位置を表す数字を含む第 2 のフィールドを有する層 $k - 1$ データ・コンテナの少なくとも 1 つのレコードを指すポインタを備え、これによって層 $k - 1$ データ・コンテナの前記レコードが、前記層 k ランクと、前記数字で表された位置との組合せによって決定された層 $k - 1$ ランクに関連付けられる、請求項 1 3 4 に記載の方法。 30

【請求項 1 3 9】

サブセットにおける識別子リストの任意の整数の位置を検索するために、層 1 データ・コンテナのレコードの第 1 のフィールドに含まれる前記データが、前記整数が前記リストの識別子を表すか示すため、それぞれが前記サブセットの個々の整数に関連付けられたビットを含むビットマップ・セグメントを備える、請求項 1 3 8 に記載の方法。

【請求項 1 4 0】

層 n ランクに関連付けられた層 n データ・コンテナの各レコードが、さらに、前記層 n ランクを含むための第 2 のフィールドを有する、請求項 1 3 8 または 1 3 9 に記載の方法。

【請求項 1 4 1】 40

$1 \leq k < n$ の場合に、各層 k データ・コンテナが、前記データ・コンテナのレコードの第 1 および第 2 のフィールドがそれぞれ記憶された少なくとも 2 つのファイルを備え、前記ファイルが別個にアクセス可能である、請求項 1 3 8 から 1 4 0 のいずれか 1 項に記載の方法。

【請求項 1 4 2】 50

$1 \leq k \leq n$ の場合に、層 k データ・コンテナの各レコードがさらに次アドレス・フィールドを有し、それによりレコード・チェーンが、次アドレス・フィールドによって層 k データ・コンテナで定義され、シソーラス項目の少なくとも一部が、それぞれ、層 n データ・コンテナのレコード・チェーンに関連付けられ、層 n で前記項目の 1 つに関連するコード化データが、層 n データ・コンテナで、自身に関連付けられるレコード・チェーンに記憶されるか、そこから検索可能である、請求項 1 3 4 から 1 4 1 のいずれか 1 項に記載の方

法。

【請求項 1 4 3】

$1 \leq k < n$ の場合に、前記シソーラス項目が、それぞれ、層 k データ・コンテナのレコード・チェーンに関連付けられ、それにより層 k で前記項目の 1 つに関連するコード化データが、層 k データ・コンテナで、自身に関連付けられるレコード・チェーンに記憶されるか、そこから検索可能である、請求項 1 4 2 に記載の方法。

【請求項 1 4 4】

$1 < k \leq n$ の場合に、層 k データ・コンテナの各レコードが、さらに、層 $k - 1$ データ・コンテナにある個々のチェーンの第 1 のレコードのアドレスを指す頭部アドレス・フィールドを有する、請求項 1 4 2 に記載の方法。

10

【請求項 1 4 5】

$1 \leq k \leq n$ の場合に、各層 k データ・コンテナが、前記データ・コンテナのレコードの第 1 のフィールドおよび次アドレス・フィールドがそれぞれ記憶された少なくとも 2 つのファイルを備え、前記ファイルが別個にアクセス可能である、請求項 1 4 2 から 1 4 4 のいずれか 1 項に記載の方法。

【請求項 1 4 6】

参照テーブルの行を指定する識別子を含む整数範囲が、複数の所定の部分に区分され、識別子リストを表すデータの少なくとも一部が、それぞれ前記部分に関連付けられた複数の記憶セクションに分散され、前記部分の 1 つに関連付けられた記憶セクションが、前記部分の識別子で構成された識別子サブリストを表すデータを含む、請求項 1 1 8 から 1 4 5

20

【請求項 1 4 7】

参照テーブルの行識別子範囲の前記部分ごとに個々の記憶ユニットが設けられて、前記部分に関連付けられた記憶セクションを受け取る、請求項 1 4 6 に記載の方法。

【請求項 1 4 8】

マクロワード・シソーラスの少なくとも一部が、それぞれ前記部分に関連付けられた複数のセクションを有し、属性および接頭語長に関連付けられたマクロワード・シソーラスの、前記部分の 1 つに関連付けられたセクションが、前記接頭語長を有し、前記部分の識別子が割り当てられた参照テーブル行の前記データ属性に指定された少なくとも 1 つのワードの対応接頭語と一致する接頭語値ごとに、個々の項目を有し、前記項目が、前記部分に関連付けられた記憶セクションから識別子サブリストを検索するためのデータを含む、請求項 1 4 7 に記載の方法。

30

【請求項 1 4 9】

少なくとも 1 つのワード・シソーラスが、それぞれ前記部分に関連付けられた複数のセクションを有し、属性に関連付けられたワード・シソーラスの、前記部分の 1 つに関連付けられたセクションが、前記部分の識別子が割り当てられた参照テーブル行の前記属性に指定されたワードごとに、個々の項目を有し、前記項目が、前記部分に関連付けられた記憶セクションから識別子サブリストを検索するためのデータを含む、請求項 1 2 1 および 1 4 8 の両方に記載の方法。

【請求項 1 5 0】

各シソーラス項目が、記憶セクションから個々の識別子サブリストを検索するデータを含むために、それぞれ前記部分に関連付けられた複数のフィールドを有する、請求項 1 4 6 または 1 4 7 に記載の方法。

40

【請求項 1 5 1】

データベース・システムで問合せを処理する方法で、それぞれがデータ属性に関連付けられた列を有する参照テーブルの個々の行を指定するため、複数の行識別子が定義され、前記行が、データ集合体で前記属性に指定された関連ワードのグループを含み、それぞれが個々の属性に関連付けられた複数のシソーラス、およびそれぞれが前記シソーラスの項目に関連付けられた参照テーブル行識別子リストを表すデータが記憶され、1 つの属性に関連付けられた各シソーラスが、前記 1 つの属性に指定できるワード・セットのサブセット

50

への区分に関して定義され、データ集合体で前記1つの属性に指定された少なくとも1つのワードを含むサブセットごとに個々の項目を有し、前記シソーラス項目に関連付けられた参照テーブル行識別子リストが、前記1つの属性に指定された前記サブセットのワードを有する参照テーブルの行に割り当てられた任意の識別子を含み、

シソーラスが、属性と、前記属性のワード長より短い接頭語長とに関連付けられた少なくとも1つのマクロワード・シソーラスを含み、それにより前記マクロワード・シソーラスが、前記接頭語長を有する共通接頭語によって始まるワードでそれぞれ構成されたサブセットへの区分に関して定義され、

－ 問合せに関連するシソーラス項目に関わる組合せを決定するため、問合せ基準を分析するステップと、

－ 前記組合せに基づいて、および前記関連シソーラス項目に関連付けられた参照テーブル行識別子リストを表す記憶データに基づいてマッピング参照テーブル業識別子リストを決定するステップと、

－ 応答を出力するために、前記マッピング行識別子リストを処理するステップとを含む方法。

【請求項152】

少なくとも1つの属性が、異なる接頭語長に関連付けられた複数のマクロワード・シソーラスを有する、請求項151に記載の方法。

【請求項153】

問合せ基準を分析するステップが、前記基準で言及された少なくとも1つの属性について、

－ 問合せ基準の前記属性について定義されたワードの少なくとも1つの範囲を選択することと、

－ 1つまたは複数のサブセットで、データ集合体の前記属性に指定された選択範囲のワードをマッピングすることとを含み、前記1つまたは複数のサブセットごとに、シソーラス項目が、選択範囲の関連項目として保持され、

マッピング行識別子リストを決定するステップが、前記選択範囲について保持された関連シソーラス項目のデータによって表された識別子リストの個々の部分をマージすることを含む、請求項151または152に記載の方法。

【請求項154】

選択範囲ごとに保持される関連シソーラスが最小数になるよう、マッピングを実行する、請求項153に記載の方法。

【請求項155】

属性に関連付けられた各シソーラスは、各サブセットが前記属性に指定できる1つのワードで、またはワード・セットの連続ワードで構成されるよう、区分に関して定義され、前記シソーラスの項目が、前記属性に指定されたワードに基づきソートされ、問合せ基準を分析するステップが、関連シソーラス項目を識別するため、少なくとも1つのシソーラス内に少なくとも1つの二分検索を備える、請求項153または154に記載の方法。

【請求項156】

シソーラスが、さらに、それぞれが1つのワードで構成されたサブセットへの区分に関して、個々の属性に関連付けられた少なくとも1つのワード・シソーラスを備える、請求項155に記載の方法。

【請求項157】

参照テーブル行の少なくとも1つにてデフォルト値が指定された属性に関連付けられた各ワード・シソーラスが、さらに、デフォルト値の項目を有し、それによって前記参照テーブル行識別子リストの1つが、デフォルト値の前記シソーラス項目に関連付けられ、前記属性に指定された前記デフォルト値を有する参照テーブル行に割り当てられた識別子を含む、請求項156に記載の方法。

【請求項158】

問合せ基準を分析するステップが、関連するシソーラス項目を含む前記組合せを、少なく

10

20

30

40

50

とも1つの葉ノードを有する木として決定することを含み、各葉ノードが、個々の属性について保持された少なくとも1つの関連シソーラス項目に対応する、請求項143から157のいずれか1項に記載の方法。

【請求項159】

前記木が、前記少なくとも1つの葉ノードおよび少なくとも1つの演算子ノードを含む複数のノードを有し、各演算子ノードが、前記木の別のノードで表された少なくとも1つの部分基準に適用されるブール演算子を表し、演算子ノードの1つが、全マッチング基準を表す根ノードである、請求項158に記載の方法。

【請求項160】

前記木のノードが、さらに、前記問合せ基準を分析するステップの前に、参照テーブル行識別子リストが決定されている少なくとも1つの事前設定ノードを含む、請求項159に記載の方法。 10

【請求項161】

前記事前設定ノードの参照テーブル行識別子リストが、前の問合せの処理時に獲得された少なくとも1つのマッチング参照テーブル行識別子リストから決定される、請求項160に記載の方法。

【請求項162】

マッチング行識別子リストを決定するステップが、前記木のノードごとに個々の識別子リストを獲得することを含み、これによって少なくとも1つの関連シソーラス項目に対応する葉ノードごとに獲得された識別子リストが、前記少なくとも1つの関連シソーラス項目に関連付けられた識別子リストの個々の部分をマージし、少なくとも1つの部分基準に適用されるブール演算子を表す演算子ノードごとに獲得される識別子リストが、前記少なくとも1つの部分基準を表すノードについて獲得された識別子リストに前記ブール演算子を適用することによって獲得され、前記マッチング行識別子リストが、根ノードについて獲得された識別子リストとして決定される、請求項159から161のいずれか1項に記載の方法。 20

【請求項163】

獲得された前記識別子リストがそれぞれ、前記参照テーブル行に割り当てられた識別子が前記獲得リストに属するか示すため、個々の参照テーブル行に指定されたビットで構成されたビットマップ・ベクトルの形式で生成される、請求項162に記載の方法。 30

【請求項164】

n個の連続するコード化層を備えるコード化方式を使用して、シソーラス項目に関連付けられた識別子リストを表すコード化データを提供し、nが少なくとも1に等しい数字であり、各層が、前記層の入力リストの整数を含む範囲をサブセットに分割するための所定のパターンを有し、前記識別子リストが、前記シソーラス項目の第1の層の入力リストであり、最終層以外の任意の層について、入力リストの少なくとも1つの整数を含む各サブセットの位置を前記層のパターンで表す整数リストが、次層の入力リストを形成し、コード化データが、層ごとに、および入力リストの少なくとも1つの整数を含むサブセットごとに、前記サブセットにおける入力リストの各整数の位置を表すデータを備え、さらに、少なくとも前記層が最終層である場合には、前記サブセットの位置を前記層のパターンで表すデータを備える、請求項162または163に記載の方法。 40

【請求項165】

各層のパターンは、整数サブセットが、同数の整数で構成された連続的インターバルであるようなパターンである、請求項164に記載の方法。

【請求項166】

前記整数の数が、層ごとに2の整数累乗である、請求項165に記載の方法。

【請求項167】

サブセットにおける入力リストの整数の位置を表す前記データが、ビットマップ・セグメントで構成される、請求項164から166のいずれか1項に記載の方法。

【請求項168】

マッチング行識別子リストを決定するステップが、前記木のノードごとに層 n の整数リストを決定することを含み、それにより葉ノードについて決定される層 n 整数リストが、コード化方式にて、前記葉ノードが対応する関連シソーラス項目で表された識別子リストのマージに関連付けられる層 n 入力リストで構成され、それによって少なくとも 1 つの部分基準に適用されるブール演算子を表す演算子ノードごとに獲得される層 n 整数リストが、前記少なくとも 1 つの部分基準を表すノードについて決定された層 n 整数リストに前記ブール演算子を適用することによって獲得され、層 n の結果リストが、根ノードについて獲得された層 n 整数リストとして決定される、請求項 164 から 167 のいずれか 1 項に記載の方法。

【請求項 169】

10

前記木のノードが、さらに、前記問合せ基準を分析するステップの前に参照テーブル行識別子リストが決定されている少なくとも 1 つの事前設定ノードを含み、前記参照テーブル行識別子リストが、コード化方式に従い、前記事前設定ノードの前記層 n 整数リストとして決定される層 n 入力リストを提供する、請求項 168 に記載の方法。

【請求項 170】

コード化方式で、コード化層 n のサブセットにおける入力リストの各整数の位置を表すコード化データが、前記整数が前記入力リストに属するか示すため、それぞれがサブセットの個々の整数に関連付けられたビットを含む層 n ビットマップ・セグメントを定義し、前記サブセットの位置を層 n パターンで表すデータが、前記層 n ビットマップ・セグメントに関連付けられた層 n 整数ランクを備え、葉ノードについて層 n 整数リストを決定するステップが、

20

- ー 層 n ビットマップ・ベクトルを論理ゼロで初期化することと、
- ー 前記葉ノードが対応する関連シソーラス項目ごとに、コード化データから層 n ランクおよび関連するビットマップ・セグメントを獲得することと、
- ー 前記層 n ランクごとに、自身に関連付けられた層 n ビットマップ・セグメントを、前記層 n ランクによって決定された位置を有する前記層 n ビットマップ・ベクトルに重畳させることとを含み、重畳が、ビットごとのブール OR 演算に従って実行され、葉ノードの前記層 n リストが、結果層 n ビットマップ・ベクトルに対応する、請求項 168 に記載の方法。

【請求項 171】

30

$n > 1$ の場合、マッチング行識別子リストを決定するステップが、 $n - 1$ から 1 へと減少する k について、前記木のノードごとに層 k 整数リストを決定することを含み、それにより葉ノードについて決定される層 k 整数リストが、層 $k + 1$ 結果リストで表される位置を有する層 k サブセットに属し、前記葉ノードが対応する関連シソーラス項目に表された識別子リストにコード化方式で関連付けられた層 k 入力リストの任意の整数で構成され、少なくとも 1 つの部分基準に適用されるブール演算子を表す演算子ノードごとに獲得される層 k 整数リストが、前記少なくとも 1 つの部分基準を表すノードについて決定される層 k 整数リストに前記ブール演算子を適用することによって獲得され、層 k 結果リストが、根ノードについて獲得された層 k 整数リストとして決定され、前記マッチング行識別子リストが、決定された層 1 結果リストに対応する、請求項 168 から 170 のいずれか 1 項に記載の方法。

40

【請求項 172】

前記木のノードが、さらに、前記問合せ基準を分析するステップの前に参照テーブル行識別子リストが決定されている少なくとも 1 つの事前設定ノードを含み、前記参照テーブル行識別子リストが、コード化方式に従い、前記事前設定ノードについて前記層 k 整数リストとして決定される層 k 入力リストを提供する、請求項 171 に記載の方法。

【請求項 173】

コード化方式で、コード化層 $k < n$ についてサブセットにおける入力リストの各整数の位置を表すコード化データが、前記整数が前記入力リストに属するか示すため、それぞれがサブセットの個々の整数に関連付けられたビットを含む層 k ビットマップ・セグメントを

50

定義し、コード化データが、さらに、前記層 k ビットマップ・セグメントに関連付けられた層 k 整数ランクを備えて、前記サブセットの位置を層 k パターンで表し、葉ノードについて層 k 整数リストを決定するステップが、

- 層 k ビットマップ・ベクトルを論理ゼロで初期化することと、
 - 前記葉ノードが対応する関連シソーラス項目ごとに、コード化データから層 k ランクを獲得することと、
 - 層 $k + 1$ 結果リストに属し、獲得された層 k ランクを選択して、関連する層 k ビットマップ・セグメントを、選択した層 k ランクによって決定された位置を有する前記層 k ビットマップ・ベクトルのセグメントに重畳させることとを含み、重畳が、ビットごとのブール OR 演算に従って実行され、
- 葉ノードの前記層 k リストが、結果層 k ビットマップ・ベクトルに対応する、請求項 171 または 172 に記載の方法。

10

【請求項 174】

$1 \leq k < n$ の場合に、少なくとも 1 つのシソーラス項目について、層 k ランクおよびそれに関連付けられた層 k ビットマップ・セグメントが、別個の第 1 および第 2 のファイルの対応するアドレスに記憶され、葉ノードについて層 k 整数リストを決定するステップが、

- 前記第 1 および第 2 のファイルのアドレスに関連付けられたレコードを有するランク・テーブルを RAM メモリに設けることと、
- 前記第 1 のファイルで選択された層 k ランクのアドレスに関連付けられたランク・テーブル・レコードに、選択された層 k ランクを書き込むことにより、ランク・テーブルを充填することと、
- 層 k ランクを含み、第 2 のファイルのアドレスに関連付けられた充填ランク・テーブルの任意のレコードについて、第 2 のファイルの前記アドレスで関連する層 k ビットマップ・セグメントを読み出して、前記層 k ランクによって欠点された位置を有する前記層 k ビットマップ・ベクトルのセグメントに、読み出した層 k ビットマップ・セグメントを重畳させることとを含む、請求項 173 に記載の方法。

20

【請求項 175】

マッチング行識別子リストを決定するステップが、さらに、 $1 < k \leq n$ になるような任意のコード化層 k について、コード化方式の層 k の入力リストとして層 k 結果リストを提供することにより獲得された層 k' 入力リストで構成された層 k' フィルタリング・リスト

30

を $k \leq k' \leq n$ について決定することを含み、

コード化方式で、コード化層 $k < n$ のサブセットにおける入力リストの各整数の位置を表すコード化データが、前記整数が前記入力リストに属するか示すため、サブセットの個々の整数に関連付けられた各ビットを含む層 k ビットマップ・セグメントを定義し、前記層 k ビットマップ・セグメントに関連付けられた層 k 整数ランクが、前記サブセットの位置を層 k パターンで表し、 $k < n$ の場合に、葉ノードについて層 k 整数リストを決定するステップが、

- ／ a ／ 層 k ビットマップ・ベクトルを論理ゼロで初期化することと、
- ／ b ／ 前記葉ノードが対応する関連シソーラス項目ごとに、コード化データから層 n ランクを獲得し、 $k' = n$ を設定することと、
- ／ c ／ 選択された層 k' ランクごとに、
- ／ c 1 ／ 選択された層 k' ランクが、層 k' フィルタリング・リストの少なくとも 1 つの整数を含むサブセットの位置を層 k' パターンで表す場合は、選択された層 k' ランクが関連付けられた層 k' ビットマップ・セグメントを獲得し、
- ／ c 2 ／ 前記層 k' ビットマップ・セグメントで表された位置を前記サブセット内に有する層 k' フィルタリング・リストの整数がある場合は、その整数について、選択された層 k' ランク、および前記層 k' ビットマップ・セグメントで表された前記位置から決定した個々の層 $k' - 1$ ランクを選択し、
- ／ c 3 ／ $k' > k + 1$ の場合は、 k' を 1 ユニットだけ減少させてステップ／ c ／を実行し、

40

50

／c 4／ $k' - 1 = k$ の場合は、選択された層 $k' - 1$ ランクが関連付けられた層 k ビットマップ・セグメントを獲得し、前記選択層 $k' - 1$ ランクによって決定された位置を有する前記層 k ビットマップ・ベクトルのセグメントに前記層 k ビットマップ・セグメントを重畳することを含み、重畳が、ビットごとのブール OR 演算に従って実行され、葉ノードの前記層 k リストが、結果である層 k ビットマップ・ベクトルに対応する、請求項 171 または 172 に記載の方法。

【請求項 176】

$1 \leq k < n$ の場合、少なくとも 1 つのシソーラス項目に関する層 k ビットマップ・セグメントが、それぞれ自身に関連付けられた層 k ランクに対応するアドレスで少なくとも 1 つの層 k ファイルに記憶され、 $1 \leq k < n$ の場合に、葉ノードについて層 k 整数リストを決定するステップが、

— 前記層 k ファイルのアドレスに関連付けられたレコードを有するランク・テーブルを RAM メモリに設けることと、
— 選択された層 k ランクに対応するアドレスに関連付けられたランク・テーブル・レコードに、選択された層 k ランクを書き込むことにより、ランク・テーブルを充填することと、

— 層 k ランクを含み、前記層 k ファイルのアドレスに関連付けられた充填ランク・テーブルの任意のレコードについて、前記アドレスにて関連する層 k ビットマップ・セグメントを読み出して、前記層 k ランクによって決定された位置を有する前記層 k ビットマップ・ベクトルのセグメントに、読み出した層 k ビットマップ・セグメントを重畳させることとを含む、請求項 175 に記載の方法。

【請求項 177】

マッチング行識別子リストを処理するステップが、問合せで指定された少なくとも 1 つの属性について、前記属性に関連付けられたシソーラスを選択することと、マッチングデータ・グラフ識別子リストとの非空白交差を有する識別子リストが関連付けられた選択シソーラスの項目を検出することを含む、請求項 151 から 176 のいずれか 1 項に記載の方法。

【請求項 178】

問合せで指定された前記属性が、異なる接頭語長に関連付けられ、記憶された $Q + 1$ 個のシソーラスを有し、 Q は少なくとも 0 に等しい整数であり、前記 $Q + 1$ 個のシソーラスがそれぞれ、 $0 \leq q \leq Q$ になるようなシソーラス・レベル・パラメータ q を有し、それにより接頭語長が、レベル・パラメータ q の減少関数であって、 $q = 0$ の場合は前記属性のワード長に対応し、前記 $Q + 1$ 個のシソーラスがそれぞれ、前記シソーラスに関連付けられた接頭語長を有する共通の接頭語で開始するワードでそれぞれ構成されたサブセットへの個々の区分に関して定義され、前記シソーラスの項目が、接頭語長に基づき記憶される、請求項 177 に記載の方法。

【請求項 179】

選択され、レベル・パラメータ $Q_A \geq 0$ を有するシソーラスで、選択されたシソーラスの項目の検出が、

／a／ $Q_A \leq q \leq Q$ の場合に、個々のレベル q 目標リスト、およびレベル q シソーラスの連続項目を含む個々のレベル q シソーラス範囲を設けるステップと、

／b／ レベル Q 目標リストを、マッチング行識別子リストで初期化し、レベル・パラメータ q を値 Q で初期化して、レベル Q シソーラス範囲の第 1 の項目を選択するステップと、

／c／ レベル q 目標リストとレベル q シソーラス範囲の選択項目に関連付けられた識別子リストとの間の交差リストを決定するステップと、

／d／ 先行ステップ／c／で決定した交差リストが空白である場合に、レベル q シソーラス範囲の別の項目を選択して、ステップ／c／を繰り返すステップと、

／e／ q が Q_A より大きい場合、

／e1／ レベル $q - 1$ 目標リストを、先行ステップ／c／で決定した交差リストと等し

く設定し、

／e 2／ レベル $q-1$ シソーラス範囲を、選択されたレベル q シソーラス項目のレベル q 接頭語で開始するレベル $q-1$ 接頭語に関するレベル $q-1$ シソーラスの項目で構成するものと設定して、レベル $q-1$ シソーラス範囲の第 1 の項目を選択し、

／e 3／ q を 1 ユニットだけ減少させて、ステップ／c／に戻るステップと、

／f／ q が Q_A と等しい場合、

／f 1／ 検出した項目に選択したレベル Q_A シソーラス項目を含み、

／f 2／ レベル Q 目標リストが、先行ステップ／c／で決定された交差リストと等しい場合は、選択されたシソーラスでの項目の検出を終了し、

／f 3／ 前記交差リストにない少なくとも 1 つの整数を含む任意の目標リストから、先行ステップ／c／で決定された交差リストの整数を除去し、 10

／f 4／ q を、目標リストが前記交差リストにない少なくとも 1 つの整数を含む最小レベル・パラメータと設定し、

／f 5／ レベル q シソーラス範囲の別の項目を選択して、ステップ／c／に戻るステップとを含む、請求項 178 に記載の方法。

【請求項 180】

$Q \geq 1$ で、レベル・パラメータ $q \geq 1$ を有する各シソーラスが、さらに、レベル q 接頭語値に与えられた項目ごとに、前記レベル q シソーラス項目のレベル q 接頭語で開始する最低または最高レベル $q-1$ 接頭語に対応するレベル $q-1$ シソーラスの項目を指定するデータを含み、ステップ／e 2／が、選択されたレベル q シソーラス項目に指定されたレベル $q-1$ シソーラス項目を選択することを含む、請求項 179 に記載の方法。 20

【請求項 181】

n 個の連続コード化層を含むコード化方式を使用して、 $0 \leq q \leq Q$ の場合に、レベル q シソーラス項目に関連付けられた識別子リストを表すコード化データを提供し、 n が少なくとも 1 に等しい数字であり、各層が、前記層の入力リストの整数を含む範囲をサブセットに分割するための所定のパターンを有し、前記識別子リストが、前記シソーラス項目の第 1 の層の入力リストであり、最終層以外の任意の層について、入力リストの少なくとも 1 つの整数を含む各サブセットの位置を前記層のパターンで表す整数リストが、次層の入力リストを形成し、

コード化データが、層ごとに、および入力リストの少なくとも 1 つの整数を含むサブセットごとに、前記サブセットにおける入力リストの各整数の位置を表すデータを備え、少なくとも前記層が最終層である場合には、前記サブセットの位置を前記層のパターンで表すデータを備え 30

各レベル q 目標リストが層 1 およびレベル q フィルタリング・リストを形成し、 $Q_A \leq q \leq Q$ の場合に、コード化方式の層 1 入力リストとして提出されて、 $n > 1$ の場合は、 $1 < k \leq n$ について個々の層 k およびレベル q フィルタリング・リストを提供し、レベル q 目標リストから提供された前記層 k およびレベル q フィルタリング・リストが、コード化方式で前記レベル q 目標リストから獲得された層 k 入力リストである、請求項 179 または 180 に記載の方法。

【請求項 182】

各層のパターンは、整数サブセットが、同数の整数で構成された連続インターバルであるようなパターンである、請求項 181 に記載の方法。 40

【請求項 183】

前記整数の数が、各層の 2 の整数累乗である、請求項 182 に記載の方法。

【請求項 184】

レベル q 目標リストと識別子リストとの間の交差リストを決定するステップ／c／が、 $k = n$ から、

／c 1／ コード化方式で前記識別子リストから獲得された層 k 入力リストと、前記レベル q 目標リストに対応する層 k およびレベル q フィルタリング・リストとの間の層 k 交差リストを計算することと、 50

／ c 2 ／ 計算した層 k 交差リストが空白である場合は、レベル q 目標リストと識別子リストとの間の前記交差リストを空白であると判断することと、

／ c 3 ／ $k = 1$ の場合は、レベル q 目標リストと識別子リストとの間の前記交差リストを、計算された層 1 交差リストであると判断することと、

／ c 4 ／ $k > 1$ の場合は、 k を 1 ユニット減少させて、ステップ／ c 1 ／ から繰り返すこととを含む、請求項 181 から 183 のいずれか 1 項に記載の方法。

【請求項 185】

コード化方式で、コード化層 $k \leq n$ のサブセットにおける入力リストの各整数の位置を表すコード化データが、前記整数が前記入力リストに属するか示すため、サブセットの個々の整数に関連付けられた各ビットを含む層 k ビットマップ・セグメントを定義し、前記サブセットの位置を層 k パターンで表すデータが、前記層 k ビットマップ・セグメントに関連付けられた層 k 整数ランクを備え、コード化方式で識別子リストから獲得された層 k 入力リストと、第 1 の層 k ビットマップ・ベクトルによって表された層 k とレベル q フィルタリング・リストとの間の層 k 交差リストを計算するステップ／ c 1 ／ が、

- ー 第 2 の層 k ビットマップ・ベクトルを論理ゼロで初期化することと、
- ー 前記識別子リストを表すコード化データから層 k ランクを獲得することと、
- ー 獲得された中から、前記層 k およびレベル q のフィルタリング・リストの少なくとも 1 つの整数を含むサブセットの位置を層 k パターンで表す任意の層 k ランクを選択して、選択された層 k ランクが関連付けられた層 k ビットマップ・セグメントを獲得し、獲得された層 k ビットマップ・セグメントを、ビットごとのブール AND 演算に従って選択された層 k ランクで決定された位置を有する第 1 の層 k ビットマップ・ベクトルのセグメントと組み合わせることにより、選択された層 k ランクによって決定された位置を有する第 2 の層 k ビットマップ・ベクトルのセグメントを決定することとを含み、

前記層 k 交差リストが、結果である第 2 の層 k ビットマップ・ベクトルに対応する、請求項 184 に記載の方法。

【請求項 186】

$1 \leq k < n$ の場合に、層 k ランク、および少なくとも 1 つのシソーラス項目について自身に関連付けられた層 k ビットマップ・セグメントが、別個の第 1 および第 2 のファイルの対応するアドレスに記憶され、コード化方式で識別子リストから獲得された層 k 入力リストと、第 1 の層 k ビットマップ・ベクトルで表された層 k およびレベル q のフィルタリング・リストとの間の層 k 交差リストを計算するステップ／ c 1 ／ が、

- ー 前記第 1 および第 2 のファイルのアドレスに関連付けられたレコードを有するランク・テーブルを RAM メモリに設けることと、
- ー 前記第 1 のファイルの前記選択層 k ランクのアドレスに関連付けられたランク・テーブル・レコードに、選択された層 k ランクを書き込むことにより、ランク・テーブルを充填することと、
- ー 層 k ランクを含み、第 2 のファイルのアドレスに関連付けられた充填ランク・テーブルの任意のレコードについて、第 2 のファイルの前記アドレスにて関連する層 k ビットマップ・セグメントを読み出し、読み出した層 k ビットマップ・セグメントを、ビットごとのブール AND 演算に従って前記層 k ランクによって決定された位置を有する第 1 の層 k ビットマップ・ベクトルのセグメントと組み合わせて、前記層 k ランクによって決定された位置を有する第 2 の層 k ビットマップ・ベクトルのセグメントを決定することとを含む、請求項 185 に記載の方法。

【請求項 187】

$n > 1$ の場合、コード化方式にて、コード化層 $k \leq n$ についてサブセットにおける入力リストの各整数の位置を表すコード化データが、前記整数が前記入力リストに属するか示すため、サブセットの個々の整数と関連付けられた各ビットを含む層 k ビットマップ・セグメントを定義し、前記層 k ビットマップ・セグメントに関連付けられた層 k 整数ランクが、前記サブセットの位置を層 k パターンで表し、 $1 \leq k \leq n$ の場合に個々の第 1 の層 k ビットマップ・ベクトルによって表された層 k およびレベル q のフィルタリング・リストに

対応するレベル q 目標リストと識別子リストとの間の交差リストを決定するステップ／ c ／が、

／ c 1／ 第 2 のビットマップ・ベクトルを論理ゼロで初期化することと、

／ c 2／ 前記識別子リストを表すコード化データから獲得された層 n ランクを選択して、 $k = n$ を設定することと、

／ c 3／ 選択された層 k ランクごとに、

／ c 3 1／ 選択された層 k ランクが、前記層 k およびレベル q のフィルタリング・リストの少なくとも 1 つの整数を含むサブセットの位置を層 k パターンで表す場合は、選択された層 k ランクが関連付けられた層 k ビットマップ・セグメントを獲得し、

／ c 3 2／ 前記層 k ビットマップ・セグメントで表された位置を前記サブセット内に有する層 k およびレベル q のフィルタリング・リストの整数がある場合は、その整数について、選択された層 k ランク、および前記層 k ビットマップ・セグメントで表された前記位置から決定した個々の層 $k - 1$ ランクを選択し、

／ c 3 3／ $k > 2$ の場合は、 k を 1 ユニットだけ減少させてステップ／ c 3／を実行し、

／ c 3 4／ $k = 2$ の場合は、選択された層 1 ランクが関連付けられた層 1 ビットマップ・セグメントを獲得し、獲得した層 1 ビットマップ・セグメントを、ビットごとのブール AND 演算に従って前記層 1 ランクで決定した位置を有する第 1 の層 1 ビットマップ・ベクトルのセグメントと組み合わせて、前記層 1 ランクによって決定された位置を有する第 2 のビットマップ・ベクトルのセグメントを決定することを含み、

前記交差リストが、結果の第 2 のビットマップ・ベクトルに対応する、請求項 1 8 1 から 1 8 3 のいずれか 1 項に記載の方法。

【請求項 1 8 8】

少なくとも 1 つのシソーラス項目の層 1 ビットマップ・セグメントが、自身に関連付けられた層 1 ランクにそれぞれ対応するアドレスで、少なくとも 1 つの層 1 ファイルに記憶され、交差リストを決定するステップ／ c ／が、

— 前記層 1 ファイルのアドレスに関連付けられたレコードを有するランク・テーブルを RAM メモリに設けることと、

— 選択された層 1 ランクに対応するアドレスに関連付けられたランク・テーブル・レコードに、ステップ／ c 3 2／で選択された層 1 ランクを書き込むことによりランク・テーブルを充填することと、

— 層 1 ランクを含み、前記層 1 ファイルのアドレスに関連付けられた充填ランク・テーブルの任意のレコードについて、前記アドレスにて関連する層 1 ビットマップ・セグメントを読み出して、読み出した層 1 ビットマップ・セグメントを、ビットごとのブール AND 演算に従って前記層 1 ランクによって決定された位置を有する第 1 の層 1 ビットマップ・ベクトルのセグメントと組み合わせて、前記層 1 ランクによって決定された位置を有する第 2 の層 1 ビットマップ・ベクトルのセグメントを決定することを含む、請求項 1 8 7 に記載の方法。

【請求項 1 8 9】

レベル q シソーラスの項目ごとに事前フィルタリング・フラグを決定することをさらに含み、前記項目が、層 n およびレベル q のフィルタリング・リストに少なくとも 1 つの整数を含むサブセットの位置を層 n パターンで表す層 n ランクを一切定義していないコード化データによって表された参照テーブル行識別子リストに関連付けられている場合、前記事前フィルタリング・フラグが第 1 の値を有し、前記層 n およびレベル q のフィルタリング・リストに対応するレベル q の目標リストと、レベル q シソーラスの項目に関連付けられた識別子リストとの間の交差リストを決定するステップ／ c ／が、前記項目について決定された事前フィルタリング・フラグが前記第 1 の値を有する場合に、前記交差リストを空白であると決定することを含む、請求項 1 8 5 から 1 8 8 のいずれか 1 項に記載の方法。

【請求項 1 9 0】

層 n およびランク q のフィルタリング・リストに少なくとも 1 つの整数を含むサブセット

の位置を層 n パターンで表す層 n ランクを定義するコード化データによって表された行識別子リストに関連付けられたレベル q シソーラスがある場合、前記層 n ランクに関連付けられた層 n ビットマップ・セグメントが獲得され、獲得された層 n ビットマップ・セグメントが、前記サブセット内で、前記層 n およびレベル q のフィルタリング・リストの整数の位置を一切表さない場合は、前記第 1 の値が、前記項目について決定された事前フィルタリング・フラグに割り当てられる、請求項 189 に記載の方法。

【請求項 191】

マッチング行識別子リストを処理するステップが、さらに、選択されたシソーラスに検出項目があれば、それに関連付けられた出力データを出力テーブルに書き込むことを含む、請求項 177 から 190 のいずれか 1 項に記載の方法。

10

【請求項 192】

出力テーブルが、マッチング行識別子リストの各識別子に対応する個々の行を含み、選択シソーラスの検出された項目に関連付けられた出力データが、マッチング行識別子リストと前記検出シソーラス項目に関連付けられた識別子リストとの両方に属する参照テーブル行識別子に対応する出力テーブルの任意の行に書き込まれる、請求項 191 に記載の方法。

【請求項 193】

各参照テーブル行識別子が、自身に対応する出力テーブルの個々の行を有し、出力テーブルの行が、出力データを書き込む前にデフォルト値で初期化され、デフォルト値を含まない出力テーブルの行が、出力データの書き込み後に読み出される、請求項 192 に記載の方法。

20

【請求項 194】

出力テーブルが、参照テーブル行識別子ごとに個々のレコードを有する索引ファイルに関連付けられ、デフォルト値、または前記参照テーブル行識別子に対応する出力テーブルの個々の行を指定するポインタを含み、索引ファイルのレコードが、出力データの書き込み前にデフォルト値で初期化され、マッチング行識別子リストと、第 1 の選択シソーラスの前記検出項目に関連付けられた識別子リストとの両方に属する参照テーブル行識別子ごとに、前記第 1 選択シソーラスの検出項目に関連付けられた出力データを書き込むステップが、

- ー 前記参照テーブル行識別子に対応して、出力テーブルの使用可能な行を割り当てることと、
- ー 出力データを割り当てた行に書き込むことと、
- ー 割り当てた行を指すポインタを、前記参照テーブル行識別子のために設けた索引ファイルのレコードに書き込むこととを含む、請求項 192 に記載の方法。

30

【請求項 195】

出力テーブルが、シソーラスが選択された個々の属性にそれぞれ関連付けられた複数の列を有し、出力テーブルの列に関連付けられた属性について選択されたシソーラスの検出項目に関連付けられた出力データが、前記列に書き込まれる、請求項 192 から 194 のいずれか 1 項に記載の方法。

【請求項 196】

少なくとも 1 つの第 2 の選択シソーラスの検出項目に関連付けられた出力データを書き込むステップが、マッチング行識別子リストと第 2 の選択シソーラスの前記検出項目に関連付けられた識別子リストとの両方に属する参照テーブル識別子ごとに、

- ー 前記参照テーブル識別子のために設けた索引ファイルのレコードに含まれたポインタを読み出すことと、
- ー 出力データを、前記ポインタによって指定された出力テーブルの行に書き込むこととを含む、請求項 194 および 195 の両方に記載の方法。

40

【請求項 197】

選択されたシソーラスが、それぞれ 1 ワードで構成されたサブセットへの区分に関して定義されたワード・シソーラスであり、検出項目に関連付けられた出力データがワードを備

50

え、そのワードのために前記検出項目が設けられている、請求項 191 から 196 のいずれか 1 項に記載の方法。

【請求項 198】

選択されたシソーラスが、接頭語長に関連付けられ、それぞれが前記接頭語長を有する共通接頭語で始まるワードで構成されたサブセットへの区分に関して定義されたマクロワード・シソーラスであり、検出項目に関連付けられた出力データが接頭語値を備え、その接頭語値のために前記検出項目が設けられている、請求項 191 から 196 のいずれか 1 項に記載の方法。

【請求項 199】

検出項目に関連付けられた出力データが、選択シソーラスの前記検出項目のアドレスを備える、請求項 191 から 196 のいずれか 1 項に記載の方法。 10

【請求項 200】

選択シソーラスの検出項目に関連付けられた出力データが、前記シソーラス項目から導出された数値を備える、請求項 191 から 196 のいずれか 1 項に記載の方法。

【請求項 201】

少なくとも 1 つの選択シソーラスの検出項目について、前記数値が、前記項目に記憶したシソーラス値に数学的関数を適用することによって計算される、請求項 200 に記載の方法。

【請求項 202】

少なくとも 1 つの選択シソーラスの検出項目について、前記数値が、前記項目に記憶されたシソーラス値、および出力テーブルに既に存在する少なくとも 1 つの値を含む複数の値に、数学的関数を適用することによって計算される、請求項 200 または 201 に記載の方法。 20

【請求項 203】

出力テーブルが、マッチング行識別子リストの各識別子に対応する個々の行を含み、検出されたシソーラス項目から導出された数値が、マッチング行識別子リストと、前記検出シソーラス項目に関連付けられた識別子リストとの両方に属する参照テーブル行識別子に対応する出力テーブルの任意の行に書き込まれる、請求項 200 から 202 のいずれか 1 項に記載の方法。

【請求項 204】

第 1 の選択シソーラスの検出項目から導出され、マッチング行識別子リストと、前記第 1 選択シソーラスの前記項目に関連付けられた識別子リストとの両方に属する参照テーブル行識別子に対応する出力テーブルの任意の行に書き込まれた数値が、前記項目に記憶されたシソーラス値から獲得され、 30

少なくとも 1 つの第 2 の選択シソーラスの検出項目から導出され、マッチング行識別子リストと、第 2 の選択シソーラスの前記項目に関連付けられた識別子リストとの両方に属する参照テーブル行識別子に対応する出力テーブルの行に書き込まれた数値が、前記項目に記憶されたシソーラス値、および出力テーブルの前記行に既に存在する少なくとも 1 つの値を含む複数の値に、数学的関数を適用することによって計算される、請求項 203 に記載の方法。 40

【請求項 205】

それぞれ出力テーブルの行に書き込まれた数値のセットから出力値を計算することをさらに含む、請求項 203 または 204 に記載の方法。

【請求項 206】

参照テーブルの行に指定された識別子を含む整数範囲が、複数の所定の部分に区分され、識別子リストを表すデータの少なくとも一部が、それぞれ前記部分に関連付けられた複数の記憶セクションに分散し、前記部分の 1 つに関連付けられた記憶セクションが、前記部分の識別子で構成された識別子サブリストを表すデータを含み、マッチング行識別子リストを決定するステップが、個々の記憶セクションによって、参照テーブル行識別子範囲の異なる部分ごとに別個に実行される、請求項 151 から 205 の 50

いずれか 1 項に記載の方法。

【請求項 207】

マッチング行識別子処理するステップが、個々の記憶セクションによって、参照テーブル行識別子範囲の異なる部分ごとに少なくとも一部は別個に実行される、請求項 206 に記載の方法。

【請求項 208】

シソーラスが、それぞれ前記部分に関連付けられた複数のセクションを有し、属性に関連付けられ、サブセットへの区分に関して定義されたシソーラスの、前記部分の 1 つに関連付けられたセクションが、前記区分のサブセットごとに、前記部分の識別子が割り当てられた参照テーブル行の前記属性に指定された少なくとも 1 つのワードを含む個々の項目を有し、前記項目が、前記属性に指定された前記サブセットのワードを有する参照テーブル行に割り当てられた前記部分の各識別子を含む識別子サブリストを表すデータを含み、問合せ基準を分析するステップが、個々のシソーラス・セクションによって、参照テーブル行識別子範囲の異なる部分ごとに、少なくとも一部は別個に実行される、請求項 206 または 207 に記載の方法。

【請求項 209】

別個のステップ実行が、参照テーブル行識別子範囲の異なる部分ごとに、個々のプロセッサによって並行して実行される、請求項 206 から 208 のいずれか 1 項に記載の方法。

【請求項 210】

各シソーラス項目が、記憶セクションから個々の識別子サブリストを検索するためのデータを含むため、前記部分に個々に関連付けられた複数のフィールドを有し、問合せ基準を分析するステップが、参照テーブル行識別子範囲の全ての部分について中央で実行され、記憶セクションによってマッチング行識別子リストを決定するステップを実行するプロセッサが使用する関連シソーラス項目が、前記記憶セクションから識別子サブリストを検索するため、データによって指定される、請求項 209 に記載の方法。

【請求項 211】

問合せ基準を分析するステップが、通信網を通して前記プロセッサに接続された問合せサーバによって実行される、請求項 210 に記載の方法。

【請求項 212】

リスト更新サーバが、通信網を通して、それぞれ前記プロセッサに結合された複数の記憶ユニットに接続され、リスト更新サーバが記憶ユニットを制御して、記憶セクションを維持する、請求項 211 に記載の方法。

【請求項 213】

前記参照テーブルが、記憶されていない仮想テーブルである、請求項 151 から 212 のいずれか 1 項に記載の方法。

【請求項 214】

データベース・システムで問合せを処理する方法であって、複数の行識別子が定義されて、データ属性に個々に関連付けられた列を有する参照テーブルの個々の行を指定し、前記行が、データ集合体内で前記属性に指定された関連ワードのグループを含み、それぞれが個々の属性に関連付けられた複数のシソーラス、およびそれぞれが前記シソーラスの項目に関連付けられた参照テーブル行識別子リストを表すデータが記憶され、1 つの属性に関連付けられた各シソーラスが、前記 1 つの属性に指定できるワード・セットのサブセットへの区分に関して定義され、データ集合体で前記 1 つの属性に指定された少なくとも 1 つのワードを含むサブセットごとに個々の項目を有し、前記シソーラス項目に関連付けられた参照テーブル行識別子リストが、前記 1 つの属性に指定された前記サブセットのワードを有する参照テーブルの行に割り当てられた任意の識別子を含み、
ー 問合せに関連するシソーラス項目の組合せ、および前記関連シソーラス項目に関連付けられた参照テーブル行識別子リストを表す記憶データに基づいて、マッチング参照テーブル行識別子リストを決定するステップと、
ー 応答を出力するため、前記マッチング行識別子リストを処理するステップとを含み、

前記マッチング行識別子処理するステップが、問合せで指定された少なくとも1つの属性について、前記属性に関連付けられたシソーラスを選択することと、マッチング行識別子リストとの非空白交差を有する識別子リストに関連付けられた選択シソーラスの項目を検出することを含む方法。

【請求項 2 1 5】

問合せで指定された前記属性が、異なる接頭語長に関連付けられ、記憶された $Q + 1$ 個のシソーラスを有し、 Q は少なくとも 0 に等しい整数であり、前記 $Q + 1$ 個のシソーラスがそれぞれ、 $0 \leq q \leq Q$ になるようなシソーラス・レベル・パラメータ q を有し、それにより接頭語長が、レベル・パラメータ q の減少関数であって、 $q = 0$ の場合は、前記属性のワード長に対応し、前記 $Q + 1$ 個のシソーラスがそれぞれ、前記シソーラスに関連付けられた接頭語長を有する共通の接頭語で開始するワードでそれぞれ構成されたサブセットへの個々の区分に関して定義され、前記シソーラスの項目が、前記接頭語長に基づき記憶される、請求項 2 1 4 に記載の方法。

【請求項 2 1 6】

レベル・パラメータ $Q A \geq 0$ を有する選択されたシソーラスで、選択されたシソーラスの項目の検出が、

／ a ／ $Q A \leq q \leq Q$ の場合に、個々のレベル q 目標リストおよびレベル q シソーラスの連続項目を含む個々のレベル q シソーラス範囲を設けるステップと、

／ b ／ レベル Q 目標リストをマッチング行識別子リストで初期化し、レベル・パラメータ q を値 Q で初期化し、レベル Q シソーラス範囲の第 1 の項目を選択するステップと、

／ c ／ レベル q 目標リストとレベル q シソーラス範囲の選択項目に関連付けられた識別子リストとの間の交差リストを決定するステップと、

／ d ／ 先行ステップ／ c ／で決定した交差リストが空白である場合に、レベル q シソーラス範囲の別の項目を選択して、ステップ／ c ／を繰り返すステップと、

／ e ／ q が $Q A$ より大きい場合、

／ e 1 ／ レベル $q - 1$ 目標リストを先行ステップ／ c ／で決定した交差リストと等しく設定し、

／ e 2 ／ レベル $q - 1$ シソーラス範囲を、選択されたレベル q シソーラス項目のレベル q 接頭語で開始するレベル $q - 1$ 接頭語に関するレベル $q - 1$ シソーラスの項目で構成するものと設定し、

／ e 3 ／ q を 1 ユニットだけ減少させて、ステップ／ c ／に戻るステップと、

／ f ／ q が $Q A$ と等しい場合、

／ f 1 ／ 検出した項目に選択したレベル $Q A$ シソーラス項目を含み、

／ f 2 ／ レベル Q 目標リストが、先行ステップ／ c ／で決定された交差リストと等しい場合は、選択されたシソーラスでの項目の検出を終了し、

／ f 3 ／ 前記交差リストにない少なくとも 1 つの整数を含む任意の目標リストから、先行ステップ／ c ／で決定された交差リストの整数を除去し、

／ f 4 ／ q を、目標リストが前記交差リストにない少なくとも 1 つの整数を含む最小レベル・パラメータと設定し、

／ f 5 ／ レベル q シソーラス範囲の別の項目を選択し、ステップ／ c ／に戻るステップとを含む、請求項 2 1 5 に記載の方法。

【請求項 2 1 7】

$Q \geq 1$ で、レベル・パラメータ $q \geq 1$ を有する各シソーラスが、さらに、レベル q 接頭語値に与えられた項目ごとに、前記レベル q シソーラス項目のレベル q 接頭語で開始する最低または最高レベル $q - 1$ 接頭語に対応するレベル $q - 1$ シソーラスの項目を指定するデータを含み、ステップ／ e 2 ／が、選択されたレベル q シソーラス項目に指定されたレベル $q - 1$ シソーラス項目を選択することを含む、請求項 2 1 6 に記載の方法。

【請求項 2 1 8】

n 個の連続コード化層を含むコード化方式を使用して、 $0 \leq q \leq Q$ の場合に、レベル q シソーラス項目に関連付けられた識別子リストを表すコード化データを提供し、 n が少なく

とも 1 に等しい数字であり、各層が、前記層の入力リストの整数を含む範囲をサブセットに分割するための所定のパターンを有し、前記識別子リストが、前記シソーラス項目の第 1 の層の入力リストであり、最終層以外の任意の層について、前記入力リストの少なくとも 1 つの整数を含む各サブセットの位置を前記層のパターンで表す整数リストが、次層の入力リストを形成し、

コード化データが、層ごとに、および前記入力リストの少なくとも 1 つの整数を含むサブセットごとに、前記サブセットにおける前記入力リストの各整数の位置を表すデータを備え、少なくとも前記層が最終層である場合には、前記サブセットの位置を前記層のパターンで表すデータを備え

各レベル q 目標リストが層 1 およびレベル q フィルタリング・リストを形成し、 $Q A \leq q \leq Q$ の場合に、コード化方式の層 1 入力リストとして提出されて、 $n > 1$ の場合は、 $1 < k \leq n$ について個々の層 k およびレベル q フィルタリング・リストを提供し、レベル q 目標リストから提供された前記層 k およびレベル q フィルタリング・リストが、コード化方式で前記レベル q 目標リストから獲得された層 k 入力リストである、請求項 2 1 6 または 2 1 7 に記載の方法。 10

【請求項 2 1 9】

各層のパターンは、整数サブセットが、同数の整数で構成された連続インターバルであるようなパターンである、請求項 2 1 8 に記載の方法。

【請求項 2 2 0】

前記整数の数が、各層の 2 の整数累乗である、請求項 2 1 9 に記載の方法。 20

【請求項 2 2 1】

レベル q 目標リストと識別子リストとの間の交差リストを決定するステップ／c／が、 $k = n$ から、

／c 1／ コード化方式で前記識別子リストから獲得された層 k 入力リストと、前記レベル q 目標リストに対応する層 k およびレベル q フィルタリング・リストとの間の層 k 交差リストを計算することと、

／c 2／ 計算した層 k 交差リストが空白である場合は、レベル q 目標リストと識別子リストとの間の前記交差リストを空白であると判断することと、

／c 3／ $k = 1$ の場合は、レベル q 目標リストと識別子リストとの間の前記交差リストを、計算された層 1 交差リストであると判断することと、 30

／c 4／ $k > 1$ の場合は、 k を 1 ユニット減少させて、ステップ／c 1／から繰り返すこととを含む、請求項 2 1 8 から 2 2 0 のいずれか 1 項に記載の方法。

【請求項 2 2 2】

コード化方式で、コード化層 $k \leq n$ のサブセットにおける入力リストの各整数の位置を表すコード化データが、前記整数が前記入力リストに属するか示すため、サブセットの個々の整数に関連付けられた各ビットを含む層 k ビットマップ・セグメントを定義し、前記サブセットの位置を層 k パターンで表すデータが、前記層 k ビットマップ・セグメントに関連付けられた層 k 整数ランクを備え、コード化方式で識別子リストから獲得された層 k 入力リストと、第 1 の層 k ビットマップ・ベクトルによって表された層 k およびレベル q フィルタリング・リストとの間の層 k 交差リストを計算するステップ／c 1／が、 40

— 第 2 の層 k ビットマップ・ベクトルを論理ゼロで初期化することと、

— 前記識別子リストを表すコード化データから層 k ランクを獲得することと、

— 前記層 k およびレベル q のフィルタリング・リストの少なくとも 1 つの整数を含むサブセットの位置を層 k パターンで表す任意の層 k ランクを選択し、選択された層 k ランクが関連付けられた層 k ビットマップ・セグメントを獲得し、獲得した層 k ビットマップ・セグメントを、ビットごとのブール AND 演算に従って選択された層 k ランクで決定された位置を有する第 1 の層 k ビットマップ・ベクトルのセグメントと組み合わせることにより、選択された層 k ランクによって決定された位置を有する第 2 の層 k ビットマップ・ベクトルのセグメントを決定することとを含む、

前記層 k 交差リストが、結果である第 2 の層 k ビットマップ・ベクトルに対応する、請求 50

項 2 2 1 に記載の方法。

【請求項 2 2 3】

$1 \leq k < n$ の場合に、層 k ランク、および少なくとも 1 つのシソーラス項目について自身に関連付けられた層 k ビットマップ・セグメントが、別個の第 1 および第 2 のファイルの対応するアドレスに記憶され、コード化方式で識別子リストから獲得された層 k 入力リストと、第 1 の層 k ビットマップ・ベクトルで表された層 k およびレベル q のフィルタリング・リストとの間の層 k 交差リストを計算するステップ / c 1 / が、

- 前記第 1 および第 2 のファイルのアドレスに関連付けられたレコードを有するランク・テーブルを RAM メモリに設けることと、
- 前記第 1 のファイルの前記選択層 k ランクのアドレスに関連付けられたランク・テーブル・レコードに、選択された層 k ランクを書き込むことにより、ランク・テーブルを充填することと、
- 層 k ランクを含み、第 2 のファイルのアドレスに関連付けられた充填ランク・テーブルの任意のレコードについて、第 2 のファイルの前記アドレスにて関連する層 k ビットマップ・セグメントを読み出して、読み出した層 k ビットマップ・セグメントを、ビットごとのブール AND 演算に従って前記層 k ランクによって決定された位置を有する第 1 の層 k ビットマップ・ベクトルのセグメントと組み合わせて、前記層 k ランクによって決定された位置を有する第 2 の層 k ビットマップ・ベクトルのセグメントを決定することを含む、請求項 2 2 2 に記載の方法。

10

【請求項 2 2 4】

20

$n > 1$ の場合、コード化方式にて、コード化層 $k \leq n$ についてサブセットにおける入力リストの各整数の位置を表すコード化データが、前記整数が前記入力リストに属するか示すため、サブセットの個々の整数と関連付けられた各ビットを含む層 k ビットマップ・セグメントを定義し、前記層 k ビットマップ・セグメントに関連付けられた層 k 整数ランクが、前記サブセットの位置を層 k パターンで表し、 $1 \leq k \leq n$ の場合に個々の第 1 の層 k ビットマップ・ベクトルによって表された層 k およびレベル q のフィルタリング・リストに対応するレベル q 目標リストと識別子リストとの間の交差リストを決定するステップ / c

- / c 1 / 第 2 のビットマップ・ベクトルを論理ゼロで初期化することと、
- / c 2 / 前記識別子リストを表すコード化データから獲得された層 n ランクを選択して、 $k = n$ を設定することと、

30

- / c 3 / 選択された層 k ランクごとに、
- / c 3 1 / 選択された層 k ランクが、前記層 k およびレベル q のフィルタリング・リストの少なくとも 1 つの整数を含むサブセットの位置を層 k パターンで表す場合は、選択された層 k ランクが関連付けられた層 k ビットマップ・セグメントを獲得し、
- / c 3 2 / 前記層 k ビットマップ・セグメントで表された位置を前記サブセット内に有する層 k およびレベル q のフィルタリング・リストの整数がある場合は、その整数について、選択された層 k ランク、および前記層 k ビットマップ・セグメントで表された前記位置から決定した個々の層 $k - 1$ ランクを選択し、

- / c 3 3 / $k > 2$ の場合は、 k を 1 ユニットだけ減少させてステップ / c 3 / を実行し

40

- / c 3 4 / $k = 2$ の場合は、選択された層 1 ランクが関連付けられた層 1 ビットマップ・セグメントを獲得し、獲得した層 1 ビットマップ・セグメントを、ビットごとのブール AND 演算に従って前記層 1 ランクで決定した位置を有する第 1 の層 1 ビットマップ・ベクトルのセグメントと組み合わせて、前記層 1 ランクによって決定された位置を有する第 2 のビットマップ・ベクトルのセグメントを決定することを含み、
- 前記交差リストが、結果の第 2 のビットマップ・ベクトルに対応する、請求項 2 1 8 から 2 2 0 のいずれか 1 項に記載の方法。

【請求項 2 2 5】

少なくとも 1 つのシソーラス項目の層 1 ビットマップ・セグメントが、自身に関連付けら

50

れた層 1 ランクにそれぞれ対応するアドレスで、少なくとも 1 つの層 1 ファイルに記憶され、交差リストを決定するステップ／c／が、

- ー 前記層 1 ファイルのアドレスに関連付けられたレコードを有するランク・テーブルを RAM メモリに設けることと、
- ー 選択された層 1 ランクに対応するアドレスに関連付けられたランク・テーブル・レコードに、ステップ／c 3 2／で選択された層 1 ランクを書き込むことにより、ランク・テーブルを充填することと、
- ー 層 1 ランクを含み、前記層 1 ファイルのアドレスに関連付けられた充填ランク・テーブルの任意のレコードについて、前記アドレスにて関連する層 1 ビットマップ・セグメントを読み出して、読み出した層 1 ビットマップ・セグメントを、ビットごとのブール AND 演算に従って前記層 1 ランクによって決定された位置を有する第 1 の層 1 ビットマップ・ベクトルのセグメントと組み合わせて、前記層 1 ランクによって決定された位置を有する第 2 の層 1 ビットマップ・ベクトルのセグメントを決定することを含む、請求項 2 2 4 に記載の方法。

10

【請求項 2 2 6】

さらに、レベル q シソーラスの項目ごとに事前フィルタリング・フラグを決定することを含み、前記項目が、層 n およびレベル q のフィルタリング・リストに少なくとも 1 つの整数を含むサブセットの位置を層 n パターンで表す層 n ランクを一切定義していないコード化データによって表された参照テーブル行識別子リストに関連付けられている場合、前記事前フィルタリング・フラグは第 1 の値を有し、前記層 n およびレベル q のフィルタリング・リストに対応するレベル q の目標リストと、レベル q シソーラスの項目に関連付けられた識別子リストとの間の交差リストを決定するステップ／c／が、前記項目について決定された事前フィルタリング・フラグが前記第 1 の値を有する場合に、前記交差リストを空白であると決定することを含む、請求項 2 2 2 から 2 2 5 のいずれか 1 項に記載の方法。

20

【請求項 2 2 7】

層 n およびランク q のフィルタリング・リストに少なくとも 1 つの整数を含むサブセットの位置を層 n パターンで位置を表す層 n ランクを定義するコード化データによって表された行識別子リストに関連付けられたレベル q シソーラスがある場合、その項目について、前記層 n ランクに関連付けられた層 n ビットマップ・セグメントが獲得され、獲得された層 n ビットマップ・セグメントが、前記サブセット内で、前記層 n およびレベル q のフィルタリング・リストの整数の位置を一切表さない場合は、前記第 1 の値が、前記項目について決定された事前フィルタリング・フラグに割り当てられる、請求項 2 2 6 に記載の方法。

30

【請求項 2 2 8】

マッチング行識別子リストを処理するステップが、さらに、選択されたシソーラスに検出項目があれば、それに関連付けられた出力データを出力テーブルに書き込むことを含む、請求項 2 1 4 から 2 2 7 のいずれか 1 項に記載の方法。

【請求項 2 2 9】

出力テーブルが、マッチング行識別子リストの各識別子に対応する個々の行を含み、選択シソーラスの検出された項目に関連付けられた出力データが、マッチング行識別子リストと、前記検出シソーラス項目に関連付けられた識別子リストとの両方に属する参照テーブル行識別子に対応する出力テーブルの任意の行に書き込まれる、請求項 2 2 8 に記載の方法。

40

【請求項 2 3 0】

各参照テーブル行識別子が、自身に対応する出力テーブルの個々の行を有し、出力テーブルの行が、出力データを書き込む前にデフォルト値で初期化され、デフォルト値を含まない出力テーブルの行が、出力データの書き込み後に読み出される、請求項 2 2 9 に記載の方法。

【請求項 2 3 1】

50

出力テーブルが、参照テーブル行識別子ごとに個々のレコードを有する索引ファイルに関連付けられ、デフォルト値、または前記参照テーブル行識別子に対応する出力テーブルの個々の行を指定するポインタを含み、索引ファイルのレコードが、出力データの書き込み前にデフォルト値で初期化され、マッチング行識別子リストと、第1の選択シソーラスの前記検出項目に関連付けられた識別子リストとの両方に属する参照テーブル行識別子ごとに、前記第1の選択シソーラスの検出項目に関連付けられた出力データを書き込むステップが、

- ー 前記参照テーブル行識別子に対応して、出力テーブルの使用可能な行を割り当てることと、
- ー 出力データを割り当てた行に書き込むことと、
- ー 割り当てた行を指すポインタを、前記参照テーブル行識別子のために設けた索引ファイルのレコードに書き込むこととを含む、請求項229に記載の方法。

10

【請求項232】

出力テーブルが、シソーラスが選択された個々の属性にそれぞれ関連付けられた複数の列を有し、出力テーブルの列に関連付けられた属性について選択されたシソーラスの検出項目に関連付けられた出力データが、前記列に書き込まれる、請求項229から231のいずれか1項に記載の方法。

【請求項233】

少なくとも1つの第2の選択シソーラスの検出項目に関連付けられた出力データを書き込むステップが、マッチングデータ・グラフ識別子リストと、第2の選択シソーラスの前記検出項目に関連付けられた識別子リストとの両方に属するデータ・グラフ識別子ごとに、

20

- ー 前記データ・グラフ識別子のために設けた索引ファイルのレコードに含まれたポインタを読み出すことと、
- ー 出力データを、前記ポインタによって指定された出力テーブルの行に書き込むこととを含む、請求項231および232の両方に記載の方法。

【請求項234】

選択されたシソーラスが、それぞれ1ワードで構成されたサブセットへの区分に関して定義されたワード・シソーラスであり、検出項目に関連付けられた出力データがワードを備え、そのワードのために前記検出項目が設けられている、請求項228から233のいずれか1項に記載の方法。

30

【請求項235】

選択されたシソーラスが、接頭語長に関連付けられ、それぞれが前記接頭語長を有する共通接頭語で始まるワードで構成されたサブセットへの区分に関して定義されたマクロワード・シソーラスであり、検出項目に関連付けられた出力データが接頭語値を備え、その接頭語値のために前記検出項目が設けられている、請求項228から233のいずれか1項に記載の方法。

【請求項236】

検出項目に関連付けられた出力データが、選択シソーラスの前記検出項目のアドレスを備える、請求項228から233のいずれか1項に記載の方法。

【請求項237】

40

選択シソーラスの検出項目に関連付けられた出力データが、前記シソーラス項目から導出された数値を備える、請求項228から233のいずれか1項に記載の方法。

【請求項238】

少なくとも1つの選択シソーラスの検出項目について、前記数値が、前記項目に記憶したシソーラス値に数学的関数を適用することによって計算される、請求項237に記載の方法。

【請求項239】

少なくとも1つの選択シソーラスの検出項目について、前記数値が、前記項目に記憶されたシソーラス値、および出力テーブルに既に存在する少なくとも1つの値を含む複数の値に、数学的関数を適用することによって計算される、請求項237または238に記載の

50

方法。

【請求項 2 4 0】

出力テーブルが、マッチング行識別子リストの各識別子に対応する個々の行を含み、検出されたシソーラス項目から導出された数値が、マッチング行識別子リストと、前記検出シソーラス項目に関連付けられた識別子リストとの両方に属する参照テーブル行識別子に対応する出力テーブルの任意の行に書き込まれる、請求項 2 3 7 から 2 3 9 のいずれか 1 項に記載の方法。

【請求項 2 4 1】

第 1 の選択シソーラスの検出項目から導出され、マッチング行識別子リストと、前記第 1 の選択シソーラスの前記項目に関連付けられた識別子リストとの両方に属する参照テーブル行識別子に対応する出力テーブルの任意の行に書き込まれた数値が、前記項目に記憶されたシソーラス値から獲得され、

少なくとも 1 つの第 2 の選択シソーラスの検出項目から導出され、マッチング行識別子リストと、第 2 の選択シソーラスの前記項目に関連付けられた識別子リストとの両方に属する参照テーブル行識別子に対応する出力テーブルの行に書き込まれた数値が、前記項目に記憶されたシソーラス値、および出力テーブルの前記行に既に存在する少なくとも 1 つの値を含む複数の値に、数学的関数を適用することによって計算される、請求項 2 4 0 に記載の方法。

【請求項 2 4 2】

それぞれ出力テーブルの行に書き込まれた数値のセットから出力値を計算することをさらに含む、請求項 2 4 0 または 2 4 1 に記載の方法。

【請求項 2 4 3】

参照テーブルの行に指定された識別子を含む整数範囲が、複数の所定の部分に区分され、識別子リストを表すデータの少なくとも一部が、それぞれ前記部分に関連付けられた複数の記憶セクションに分散し、前記部分の 1 つに関連付けられた記憶セクションが、前記部分の識別子で構成された識別子サブリストを表すデータを含み、マッチング行識別子リストを決定するステップが、個々の記憶セクションによって、参照テーブル行識別子範囲の異なる部分ごとに別個に実行される、請求項 2 1 4 から 2 4 2 のいずれか 1 項に記載の方法。

【請求項 2 4 4】

マッチング行識別子処理するステップが、個々の記憶セクションによって、参照テーブル行識別子範囲の異なる部分ごとに、少なくとも一部は別個に実行される、請求項 2 4 3 に記載の方法。

【請求項 2 4 5】

別個のステップ実行が、参照テーブル行識別子範囲の異なる部分ごとに、個々のプロセッサによって並行して実行される、請求項 2 4 3 または 2 4 4 に記載の方法。

【請求項 2 4 6】

問合せに関連するシソーラス項目の組合せを、通信網を通して前記プロセッサに接続された問合せサーバによって、問合せで指定された基準に基づき決定する、請求項 2 4 5 に記載の方法。

【請求項 2 4 7】

リスト更新サーバが、通信網を通して、それぞれ前記プロセッサに結合された複数の記憶ユニットに接続され、リスト更新サーバが記憶ユニットを制御して、記憶セクションを維持する、請求項 2 4 6 に記載の方法。

【請求項 2 4 8】

前記参照テーブルが、記憶されていない仮想テーブルである、請求項 2 1 4 から 2 4 7 のいずれか 1 項に記載の方法。

【請求項 2 4 9】

コンピュータ・システムで整数リストを符号化する方法であって、

一 入力リストの整数を含む範囲を、所定のパターンに従ってサブセットに分割するステ

10

20

30

40

50

ップと、

ー 入力リストに少なくとも1つの整数を含むサブセットごとに、前記サブセットの位置をパターンで表すデータ、および前記サブセットにおける入力リストの各整数の位置を表すデータを含むコード化データを生成するステップとを含む方法。

【請求項 250】

サブセットにおける入力リストの各整数の位置を表すデータが、前記整数が入力リストに属するか示すため、それぞれがサブセットの個々の整数に関連付けられたビットを含むビットマップ・セグメントで構成される、請求項 249 に記載の方法。

【請求項 251】

パターンにおける各サブセットの位置が、前記サブセットが入力リストに少なくとも1つの整数を含む場合、対応するビットマップ・セグメントに関連して、コード化データに含まれる整数ランクで表される、請求項 250 に記載の方法。 10

【請求項 252】

個々のアドレスを有するレコードを備えるコード化データ・コンテナが、複数の整数リストから生成されたコード化データをまとめて記憶するために設けられ、コード化データ・コンテナの各レコードが、パターンに関連する整数ランクを記憶するための第1のフィールド、アドレス値を記憶する第2のフィールド、およびビットマップ・セグメントを記憶する第3のフィールドを含み、非空白入力リストを符号化することが、

a / コード化データ・コンテナの使用可能なレコードを選択するステップと、
b / レコードが割り当てられていない入力リストの少なくとも1つの整数を含むサブセ 20
ットを選択するステップと、

c / 選択レコードを選択サブセットに割り当てるステップと、

d / 選択サブセットに生成されたコード化データのランクおよびビットマップ・セグメントを、それぞれ選択レコードの第1および第3のフィールドに記憶するステップと、

e / 入力リストに少なくとも1つの整数を含むどのサブセットも、自身に割り当てられたレコードを有する場合は、選択レコードの第2のフィールドに終了値を記憶するステップと、

f / 入力リストに少なくとも1つの整数を含む少なくとも1つのサブセットに、自身に割り当てられたレコードがない場合は、コード化データ・コンテナの使用可能なレコードのアドレスを選択レコードの第2のフィールドに記憶し、前記使用可能なレコードを選択 30
して、ステップ b / から繰り返すステップとを含む請求項 251 に記載の方法。

【請求項 253】

コード化データ・コンテナが、レコードの第1および第2のフィールドを備える第1のファイルと、レコードの第3のフィールドを備える第2のファイルとを有し、第1および第2のファイルが別個にアクセス可能である、請求項 252 に記載の方法。

【請求項 254】

さらに、符号化された整数リストについてサブセットに割り当てられたレコードが、連続アドレスを有するように、データ・コンテナに記憶されたレコードをグループ分けするステップを含む、請求項 252 または 253 に記載の方法。

【請求項 255】

1つの整数リストから生成したコード化データが、前記1つの整数リストに割り当てられた少なくとも1つのファイルに記憶される、請求項 249 から 252 のいずれか1項に記載の方法。 40

【請求項 256】

コード化データが、共通アドレス指定を有する第1および第2のファイルに記憶され、入力リストに少なくとも1つの整数を含むサブセットごとに、前記サブセットの位置をパターンで表すデータが第1のファイルに記憶され、前記サブセットにおける入力リストの各整数の位置を表すデータが、第2のファイルの対応するアドレスに記憶される、請求項 255 に記載の方法。

【請求項 257】

サブセットが、同数の整数で構成された連続インターバルである、請求項 2 4 9 から 2 5 6 のいずれか 1 項に記載の方法。

【請求項 2 5 8】

前記整数の数が 2 の整数累乗である、請求項 2 5 7 に記載の方法。

【請求項 2 5 9】

コンピュータ・システムで整数リストを符号化する方法であって、 n 個の連続コード化層を備え、 n が少なくとも 1 に等しい数字であり、各コード化層が、

ー 前記層の入力リストの整数を含む範囲を、所定のパターンに従ってサブセットに分割するステップと、

ー 入力リストに少なくとも 1 つの整数を含むサブセットごとに、前記サブセットにおける入力リストの各整数の位置を表すデータ、および前記層が最終コード化層である場合は、前記サブセットの位置をパターンで表すデータを含むコード化データを生成するステップと、

ー 前記層が最終コード化層ではない場合に、入力リストに少なくとも 1 つの整数を含む各サブセットの位置を前記層のパターンで表すもう 1 つの整数リストを形成し、前記もう 1 つの整数リストを次層の入力リストとして提供するステップとを含む方法。

【請求項 2 6 0】

各層のパターンにおいて、サブセットが、同数の整数で構成された連続インターバルである、請求項 2 5 9 に記載の方法。

【請求項 2 6 1】

前記整数の数が、層ごとに 2 の整数累乗である、請求項 2 6 0 に記載の方法。

【請求項 2 6 2】

層ごとに生成されるコード化データが、共通アドレス指定を有する第 1 および第 2 のファイルに記憶され、前記層の入力リストに少なくとも 1 つの整数を含むサブセットごとに、前記サブセットの位置をパターンで表すデータが、第 1 のファイルに記憶され、前記サブセットにおける入力リストの各整数の位置を表すデータが、第 2 のファイルの対応するアドレスに記憶される、請求項 2 5 9 から 2 6 1 のいずれか 1 項に記載の方法。

【請求項 2 6 3】

第 1 の層に入力された 1 つの整数リストから生成されたコード化データが、前記 1 つの整数リストに割り当てられた少なくとも 1 つのファイルに記憶される、請求項 2 5 9 から 2 6 2 のいずれか 1 項に記載の方法。

【請求項 2 6 4】

第 1 の層に入力された 1 つの整数リストから生成されたコード化データが、複数の整数リストに割り当てられたデータ・コンテナに、少なくとも 1 つのレコード・チェーンとして記憶される、請求項 2 5 9 から 2 6 2 のいずれか 1 項に記載の方法。

【請求項 2 6 5】

さらに、各チェーンのレコードが連続アドレスを有するよう、データ・コンテナのレコードをグループ分けするステップを含む、請求項 2 6 4 に記載の方法。

【請求項 2 6 6】

$n \geq 2$ であり、それぞれ複数のレコードを有する層 k のデータ・コンテナが、 $1 \leq k \leq n$ の場合にコンピュータ・メモリに提供され、層 k のデータ・コンテナの各レコードが、サブセットの位置を層 k のパターンで表す層 k の整数ランクに割り当てられ、サブセットの位置を層 k パターンで表す層 k の整数ランクに関連付けられた層 k のデータ・コンテナの各レコードが、層 1 入力リストに関する層 k 入力リストの任意の整数の前記サブセットにおける位置を検索するデータを含むため、第 1 のフィールドを有し、それにより前記層 k ランクと、前記第 1 のフィールドに含まれたデータから検索可能な任意の位置との組合せが、 $k > 1$ の場合は、層 $k - 1$ データ・コンテナの個々のレコードが関連付けられた層 $k - 1$ のランクを、 $k = 1$ の場合は、前記層 1 入力リストの識別子を決定する、請求項 2 5 9 から 2 6 5 のいずれか 1 項に記載の方法。

【請求項 2 6 7】

層 n ランクに関連付けられた層 n データ・コンテナの各レコードが、さらに、前記層 n ランクを含むための第 2 のフィールドを有する、請求項 266 に記載の方法。

【請求項 268】

$1 \leq k \leq n$ の場合、サブセットにおける層 k 入力リストの任意の整数の位置を検索するため、層 k データ・コンテナのレコードの第 1 のフィールドに含まれる前記データが、前記整数が前記層 k 入力リストに属するか示すため、前記サブセットの個々の整数に関連付けられた各ビットを含むビットマップ・セグメントを備える、請求項 266 または 267 に記載の方法。

【請求項 269】

$1 \leq k \leq n$ の場合、層 k ランクに関連付けられた層 k データ・コンテナの各レコードが、さらに、前記層 k ランクを含むための第 2 のフィールドを有する、請求項 268 に記載の方法。 10

【請求項 270】

各データ・コンテナが、前記データ・コンテナのレコードの第 1 および第 2 のフィールドがそれぞれ記憶されている少なくとも 2 つのファイルを備え、前記ファイルが別個にアクセス可能である、請求項 269 に記載の方法。

【請求項 271】

$1 \leq k \leq n$ の場合、層 k データ・コンテナの各レコードが、さらに、層 $k+1$ パターンのサブセットにおける層 $k+1$ 入力リストの整数の位置を表す数字を含むために、第 2 のフィールドを有し、 20

$1 \leq k \leq n$ の場合、層 k パターンのサブセットにおける層 k 入力リストの任意の整数の位置を検索するために、層 k ランクと関連付けられた層 k データ・コンテナのレコードの第 1 のフィールドに含まれた前記データが、層 k パターンの前記サブセットにおける前記層 k 入力リストの整数の位置を表す数字を含む第 2 のフィールドを含む層 $k-1$ データ・コンテナの少なくとも 1 つのレコードを指すポインタを備え、それによって層 $k-1$ データ・コンテナの前記レコードが、前記層 k ランクと、前記数字によって表された位置との組合せによって決定された層 $k-1$ ランクに関連付けられる、請求項 266 または 267 に記載の方法。

【請求項 272】

サブセットにおける層 1 入力リストの任意の整数の位置を検索するため、層 1 データ・コンテナのレコードの第 1 のフィールドに含まれる前記データが、前記整数が前記層 1 入力リストに属するか示すため、それぞれが前記サブセットの個々の整数に関連付けられたビットを含むビットマップ・セグメントを備える、請求項 271 に記載の方法。 30

【請求項 273】

$1 \leq k < n$ の場合、各層 k データ・コンテナが、前記データ・コンテナのレコードの第 1 および第 2 のフィールドがそれぞれ記憶されている少なくとも 2 つのファイルを備え、前記ファイルが別個にアクセス可能である、請求項 271 または 272 に記載の方法。

【請求項 274】

$1 \leq k \leq n$ の場合、層 k データ・コンテナの各レコードがさらに次アドレス・フィールドを有し、それによりレコード・チェーンが、次アドレス・フィールドによって層 k データ・コンテナ内で定義され、層 1 入力リストの少なくとも幾つかは、層 n データ・コンテナにあるレコード・チェーンにそれぞれ関連付けられ、それにより前記層 1 入力リストの 1 つに関する層 n のコード化データが、層 n データ・コンテナ内で自身に関連付けられたレコード・チェーンに記憶されるか、そこから検索可能である、請求項 266 から 273 のいずれか 1 項に記載の方法。 40

【請求項 275】

$1 \leq k < n$ の場合、前記層 1 入力リストが、層 k データ・コンテナのレコード・チェーンとそれぞれ関連付けられ、それにより層 k について前記層 1 入力リストの 1 つに関するコード化データが、層 k データ・コンテナ内で自身に関連付けられたレコード・チェーンに記憶されるか、そこから検索可能である、請求項 274 に記載の方法。 50

【請求項 276】

1 $< k \leq n$ の場合、層 k データ・コンテナの各レコードが、さらに、層 $k - 1$ データ・コンテナにある個々のチェーンの第 1 のレコードのアドレスを指すため、頭部アドレス・フィールドを有する、請求項 24 に記載の方法。

【請求項 277】

1 $\leq k \leq n$ の場合、各層 k データ・コンテナが、前記データ・コンテナのレコードの第 1 のフィールドおよび次アドレス・フィールドがそれぞれ記憶された少なくとも 2 つのファイルを備え、前記ファイルが別個にアクセス可能である、請求項 274 から 276 のいずれか 1 項に記載の方法。

【請求項 278】

さらに、各チェーンのレコードが連続アドレスを有するよう、各コード化層のデータ・コンテナのレコードをグループ分けするステップを含む、請求項 274 から 277 のいずれか 1 項に記載の方法。

【請求項 279】

複数の第 1 の整数リストを第 2 の整数リストに組み合わせるコンピュータ化した方法であって、第 1 の整数リストの少なくとも 1 つが、 n 個の連続コード化層を備えるコード化方式によって提供され、記憶されたコード化データで表され、 n が少なくとも 1 に等しい数字であり、各層が、前記層の入力リストの整数を含む範囲をサブセットに分割するために所定のパターンを有し、前記第 1 の整数リストが、第 1 の層の入力リストであり、最終層以外の任意の層について、入力リストに少なくとも 1 つの整数を含む各サブセットの位置を前記層のパターンで表す整数リストが、次層の入力リストを形成し、第 1 の整数リストを表す記憶コード化データが、層ごとに、および入力リストに少なくとも 1 つの整数を含むサブセットごとに、前記サブセットにおける入力リストの各整数の位置を表すデータを備え、さらに、少なくとも前記層が最終層である場合は、前記サブセットの位置を前記層のパターンで表すデータを備え、方法が、

— それぞれが第 1 の整数リストの少なくとも 1 つと対応する中間リストの組合せを定義するステップと、

— n から 1 へと減少する k について、前記組合せに従って、複数の層 k 中間リストを組み合わせることによって、層 k 結果リストを計算するステップと、

— 層 1 結果リストとして第 2 の整数リストを生成するステップとを含み、

記憶したコード化データで表される少なくとも 1 つの第 1 の整数リストに対応する任意の中間リストについて、層 n 中間リストが、前記記憶コード化データから、コード化方式で前記少なくとも 1 つの第 1 の整数リストに関連付けられた任意の層 n 入力リストの整数で構成されたものとして決定され、 $n > 1$ の場合は、 $k < n$ について各層 k 中間リストが、前記記憶コード化データおよび層 $k + 1$ 結果リストから、層 $k + 1$ 結果リストにて表された位置を有する層 k サブセットに属するコード化方式にて、前記少なくとも 1 つの第 1 の整数リストに関連付けられた層 k 入力リストの任意の整数を構成するものとして決定される方法。

【請求項 280】

各層のパターンにおいて、サブセットが、同数の整数で構成された連続インターバルである、請求項 279 に記載の方法。

【請求項 281】

前記整数の数が、層ごとに 2 の整数累乗である、請求項 280 に記載の方法。

【請求項 282】

コード化方式において、コード化層 n のサブセットにおける入力リストの各整数の位置を表すコード化データが、前記整数が前記入力リストに属するか示すため、それぞれサブセットの個々の整数に関連付けられたビットを含む層 n ビットマップ・セグメントを定義し、前記サブセットの位置を層 n パターンで表すデータが、前記層 n ビットマップ・セグメントに関連付けられた層 n 整数ランクを備え、記憶されたコード化データで表される少なくとも 1 つの第 1 の整数リストに対応する中間リストの層 n 中間リストが、

ー 層 n ビットマップ・ベクトルを論理ゼロで初期化することと、
 ー 前記記憶コード化データから層 n ランクおよび関連のビットマップ・セグメントを獲得することと、
 ー 前記層 n ランクごとに、自身に関連付けられた層 n ビットマップ・セグメントを、前記層 n ランクによって決定された位置を有する前記層 n ビットマップ・ベクトルのセグメントに重畳することを含む手順によって決定され、重畳が、ビットごとのブール OR 演算に従って、実行され、
 前記層 n 中間リストが、結果層 n ビットマップ・ベクトルに対応する、請求項 279 から 281 のいずれか 1 項に記載の方法。

【請求項 283】

$n > 1$ であり、コード化方式において、コード化層 $k < n$ についてサブセットにおける入力リストの各整数の位置を表すコード化データが、前記整数が前記入力リストに属するか示すため、それぞれサブセットの個々の整数に関連付けられたビットを含む層 k ビットマップ・セグメントを定義し、コード化データが、さらに、前記サブセットの位置を層 k パターンで表すため、前記層 k ビットマップ・セグメントに関連付けられた層 k 整数ランクを備え、 $k < n$ の場合に、記憶されたコード化データで表される少なくとも 1 つの第 1 の整数リストに対応する中間リストの層 k 中間リストが、

ー 層 k ビットマップ・ベクトルを論理ゼロで初期化することと、
 ー 前記記憶コード化データから層 k ランクを獲得することと、
 ー 層 $k + 1$ 結果リストに属し、獲得された層 k ランクを選択し、関連する層 k ビットマップ・セグメントを、選択層 k ランクによって決定された位置を有する前記層 k ビットマップ・ベクトルのセグメントに重畳することを含む手順によって決定され、前記重畳が、ビットごとのブール OR 演算に従って、実行され、
 前記層 k 中間リストが、結果層 k ビットマップ・ベクトルに対応する、請求項 282 に記載の方法。

【請求項 284】

$1 \leq k < n$ の場合に、層 k ランクおよび自身に関連付けられた層 k ビットマップ・セグメントが、別個の第 1 および第 2 のファイルの対応するアドレスに記憶され、記憶コード化データで表される少なくとも 1 つの第 1 の整数リストに対応する中間リストについて、層 k 中間リストを決定する前記手順が、

ー 前記第 1 および第 2 のファイルのアドレスに関連付けられたレコードを有するランク・テーブルを RAM メモリに設けることと、
 ー 前記第 1 のファイルの前記選択層 k ランクのアドレスに関連付けられたランク・テーブル・レコードに、選択された層 k ランクを書き込むことにより、ランク・テーブルを充填することと、
 ー 層 k ランクを含み、第 2 のファイルのアドレスに関連付けられた充填ランク・テーブルの任意のレコードについて、第 2 のファイルの前記アドレスにて関連する層 k ビットマップ・セグメントを読み出して、読み出した層 k ビットマップ・セグメントを、前記層 k ランクによって決定された位置を有する前記層 k ビットマップ・ベクトルのセグメントに重畳することを含む、請求項 283 に記載の方法。

【請求項 285】

$n > 1$ であり、 $1 < k \leq n$ になるような任意のコード化層 k について、 $k \leq k' \leq n$ の場合に層 k' フィルタリング・リストが決定され、前記層 k' フィルタリング・リストが、層 k 結果リストをコード化方式の層 k の入力リストとして提供することにより獲得され、コード化方式で、コード化層 $k < n$ のサブセットにおける入力リストの各整数の位置を表すコード化データが、前記整数が前記入力リストに属するか示すため、それぞれがサブセットの個々の整数に関連付けられたビットを含む層 k ビットマップ・セグメントを定義し、前記層 k ビットマップ・セグメントに関連付けられた層 k 整数ランクが、前記サブセットの位置を層 k パターンで表し、 $k < n$ の場合に、記憶コード化データによって表される少なくとも 1 つの第 1 の整数リストに対応する中間リストの層 k 中間リストが、

／ a ／ 層 k ビットマップ・ベクトルを論理ゼロで初期化することと、
 ／ b ／ 前記記憶コード化データから獲得された層 n ランクを選択し、 $k' = n$ を設定することと、
 ／ c ／ 選択された層 k' ランクごとに、
 ／ c 1 ／ 選択された層 k' ランクが、層 k' フィルタリング・リストの少なくとも 1 つの整数を含むサブセットの位置を層 k' パターンで表す場合は、選択された層 k' ランクが関連付けられた層 k' ビットマップ・セグメントを獲得し、
 ／ c 2 ／ 前記層 k' ビットマップ・セグメントで表された位置を前記サブセット内に有する層 k' フィルタリング・リストの整数がある場合は、その整数について、選択された層 k' ランク、および前記層 k' ビットマップ・セグメントで表された前記位置から決定した個々の層 $k' - 1$ ランクを選択し、
 ／ c 3 ／ $k' > k + 1$ の場合は、 k' を 1 ユニットだけ減少させてステップ／ c ／を実行し、
 ／ c 4 ／ $k' - 1 = k$ の場合は、選択された層 $k' - 1$ ランクが関連付けられた層 k ビットマップ・セグメントを獲得し、前記選択層 $k' - 1$ ランクによって決定された位置を有する前記層 k ビットマップ・ベクトルのセグメントに前記層 k ビットマップ・セグメントを重畳することを含む手順で決定され、重畳が、ビットごとのブール OR 演算に従って実行され、
 前記層 k 中間リストが層 k ビットマップ・ベクトルに対応する、請求項 282 に記載の方法。

10

20

30

40

50

【請求項 286】

$1 \leq k < n$ の場合、層 k ビットマップ・セグメントが、自身に関連付けられた層 k ランクにそれぞれ対応するアドレスで少なくとも 1 つの層 k ファイルに記憶され、記憶コード化データで表される少なくとも 1 つの第 1 の整数リストに対応する中間リストの層 k 中間リストを決定する前記手順が、
 ー 前記層 k ファイルのアドレスに関連付けられたレコードを有するランク・テーブルを RAM メモリに設けることと、
 ー 選択された層 k ランクに対応するアドレスに関連付けられたランク・テーブル・レコードに、選択された層 k ランクを書き込むことにより、ランク・テーブルを充填することと、
 ー 層 k ランクを含み、前記層 k ファイルのアドレスに関連付けられた充填ランク・テーブルの任意のレコードについて、前記アドレスにて関連する層 k ビットマップ・セグメントを読み出して、前記層 k ランクによって決定された位置を有する前記層 k ビットマップ・ベクトルのセグメントに、読み出した層 k ビットマップ・セグメントを重畳させることとを含む、請求項 285 に記載の方法。

【請求項 287】

複数の前記第 1 の整数リストの層 k コード化データをまとめて記憶するため、 $k \leq n$ のコード化層ごとに、個々のアドレスを有するレコードを備えるコード化データ・コンテナを設け、各層のコード化データ・コンテナの各レコードが、前記層のパターンに関連するランクを記憶するための第 1 のフィールド、アドレス値を記憶するための第 2 のフィールド、およびビットマップ・セグメントを記憶するための第 3 のフィールドを含み、それぞれによって前記アドレス値が、同じ第 1 の整数リストに関するさらなる層 k コード化データが記憶されるデータ・コンテナの別のレコードを指すか、コード化データの最後を指定する、請求項 294 から 284 のいずれか 1 項に記載の方法。

【請求項 288】

コード化層 k ごとにデータ・コンテナに記憶されるレコードが、任意の第 1 の整数リストの層 k コード化データが記憶されているレコードが連続アドレスを有するようグループ分けされる、請求項 287 に記載の方法。

【請求項 289】

コード化データが、1 つの第 1 の整数リストに割り当てられた少なくとも 1 つのファイル

に記憶される、請求項 279 から 286 のいずれか 1 項に記載の方法。

【請求項 290】

各層のコード化データが、共通アドレス指定を有する第 1 および第 2 のファイルに記憶され、それにより、前記層の入力リストの少なくとも 1 つの整数を含むサブセットごとに、前記サブセットの位置をパターンで表すデータが、第 1 のファイルに記憶され、前記サブセットにおける入力リストの各整数の位置を表すデータが、第 2 のファイルの対応するアドレスに記憶される、請求項 289 に記載の方法。

【請求項 291】

中間リストが少なくとも 1 つの事前設定リストを含み、前記事前設定リストが、コード化方式に従い $1 \leq k \leq n$ について層 k 入力リストが事前に決定されている第 1 の整数リストの 1 つで構成され、前記層 k 入力リストが、前記事前設定リストに対応する個々の層 k 中間リストである、請求項 279 から 290 のいずれか 1 項に記載の方法。 10

【請求項 292】

請求項 1 から 248 のいずれか 1 項に記載の方法を実行するため構成され、プログラムされた手段を備える、データ集合体からの情報を管理するデータベース・システム。

【請求項 293】

請求項 249 から 291 のいずれか 1 項に記載の方法を実行するため構成され、プログラムされた手段を備えるコンピュータ・システム。

【請求項 294】

コンピュータ・プログラム製品であって、前記製品を実行する場合に、請求項 1 から 291 のいずれか 1 項に記載の方法を実行するためのソフトウェア・コード部分を備える、デジタル・コンピュータの内部メモリにロード可能なコンピュータ・プログラム製品。 20

【発明の詳細な説明】

【技術分野】

【0001】

本発明は、リレーショナル・データベース管理システム（RDBMS）に関し、特に、大量のデータを記憶し、これにアクセスするためのコンピュータ化されたシステムに関する。

【背景技術】

【0002】

非制限的な方法で、本発明は、「データ・ウェアハウス」に適用することができる。銀行の出納取引や航空座席予約などのオンライン取引処理（OLTP）システムは、例えば、従業員 123124 に関する情報を検索するような特定のキーに関連したレコードを検索するよう最適化されている。これに対して、データ・ウェアハウスは、1 組のレコードを非常に迅速に検索するよう最適化されている。これは、典型的な問合せが、「地域および地区ごとに全売上げを検索する」または「1 カ月ごとに最大量のスポーツウェアを販売している店を検索する」または「去年の各製品カテゴリについて、上位 5 店舗を選択する」という理由によるものである。このような問合せは、通常、データ・テーブル内の大きい行セットにアクセスしなければならない。問合せ処理の課題は、全てまたは大部分のデータベースを線走査せずにこれらの問合せを処理することである。 40

【0003】

この問題に取り組むため、主に 5 つの方法が提案されている。（i）多次元アレイ、（ii）特殊索引、（iii）テーブル・キャッシング、（iv）最適化した外部キー結合、および（v）近似法である。

【0004】

（i）多次元アレイ（即ち、マトリクス）

この戦略は、多次元アレイまたはマトリクスとしてデータ・ウェアハウスを実装することで構成される。その例は、米国特許第 5,359,724 号および第 5,864,857 号で見ることができる。各次元がデータの属性に対応する。例えば、売上げテーブルは、店舗の場所、製品のタイプ、顧客 ID などの座標もマトリクスとして見ることができる。 50

特定の売上げは、これらの属性を全て指定することにより識別することができる。この戦略は、小さいデータベースまたは非常に密なデータベースでうまく働く。密とは、可能な値のデカルト積に全て意味がある、例えば、各顧客が全店舗から全製品を購入する可能性が高いということの意味する。これが真であることは稀なので、この方式は、希薄な (sparse) 値を扱うよう修正しなければならない。これは、希薄な属性と密な属性の概念を定義することによって実行することができる。したがって、例えば、全ての店舗が全ての製品を置いているかもしれない (マトリクスに記憶できる密な関係) が、任意の顧客にとっては、その組合せの一部しか有効でない。したがって、従来の索引は、顧客の売上げに関わっているが、店舗全体または全製品の売上げに関する問合せの密な関係に常に使用される。

10

【0005】

(ii) 特殊索引

ビットマップ検索は、データ・ウェアハウスに合わせて調整された索引構造である (例えば、米国特許第 5, 903, 888 号参照)。これらの索引は、既に、問合せの処理を加速するため一部の市販製品で使用されている。属性のビットマップ索引は、その最も簡単な形式では、属性値当たり 1 つのビット・ベクトル (すなわち、ビットマップ) で構成され、各ビットマップのサイズは、索引が付いた関係にあるレコードの数と等しい。例えば、属性が曜日の場合、その属性には曜日ごとに 1 つずつ、7 個のビットマップ・ベクトルがある。月曜日に対応するビットマップ・ベクトルは、レコード i が曜日属性に「月曜日」を含む場合、位置 i に 1 を有する。このような 1 つの値に基づく方法は、値リスト索引と呼ばれる。他の技術 (例えば、米国特許第 5, 761, 652 号) は、ビット・ベクトルをある値の範囲に関連付け、したがって給料の属性は、0 から 20, 000 ユーロ、20, 000. 01 から 35, 000 ユーロなどの範囲のベクトルになる。さらに、各ビット・ベクトルを任意の位置にあるビット値 (1 または 0) に関連付ける技術もある。したがって、属性に n ビット番号がある場合は、 $2n$ 個のビット・ベクトルがある (位置 1、ビット値 1; 位置 1、ビット値 0; 位置 2、ビット 1; ...)。

20

【0006】

ビット・ベクトルの利点は、1 つの問合せに応答するのに、複数のビット・ベクトルを容易に使用できることである。それぞれが検索付けられている幾つかの述語に関する問合せを考察してみる。従来の大部分のデータベース管理システムは、索引を 1 つ (最も「選択的」な索引であり、したがって返す行が最も少ない) しか使用しないが、複数の索引のレコード識別子と交差するよう試みるシステムもある。

30

【0007】

ビットマップの方が働きが優れているのは、この方がコンパクトであり、レコード識別子の幾つかの集まりと交差するより幾つかのビットマップと交差する方が、はるかに高速だからである。最善のケースでは、改良点が機械のワード・サイズに比例する。例えば、ワード・サイズが 32 ビットとする。この場合、一度に 2 つのビット・ベクトルが 32 ビットと交差することができる。32 ビットの各セットが、交差する 32 のレコード識別子に対応する。このような最善のケースは、各述語が選択不可能である (すなわち、多くのレコードが各述語値に一致する) が、述語全体をまとめると非常に選択的である場合に生じる。例えば、「ブラウンの髪で、眼鏡をかけ、30 歳と 40 歳の間で、青い目、コンピュータ産業で働き、カリフォルニア在住・・・の人を検索せよ」という問合せを考察された。

40

【0008】

したがって、マトリクスが最善になるのは、述語のセットが密である (デカルト積の全ての値、またはほとんど全ての値が可能である) 場合であり、ビットマップが最善になるのは、述語が密でもなく、個々に選択的でもない場合である。中間の方法 (マトリクスに十分な密度があるが、デカルト積の多くの値が存在する場合) は、多次元索引を使用する。4 分木、R 木およびその後継者などの多次元索引は、多次元スペースの可変サイズ・グリッドとして実装される。グリッドが可変サイズであるのは、ハイパースペースでは、場所

50

によってポイントの人口が異なるからである。直感的に、フランスの等人口長方形の地図を考察されたい。この長方形は、アルプスよりパリが密になる。このような索引は、空間データ（緯度・経度カドラントに含まれるポイントの検索に使用する場合）にうまく働く。しかし、この代替法は、地理的問合せを除いて商業領域ではそれほど研究されていない。この方式は、次元数の増加とともに規模拡大せず、商用システムは、通常、3次元よりはるかに多いからである。

【0009】

(i i i) テーブル・キャッシング

(そのシステムの作成者ではないので) データベース・システムの頂部に新しい索引を設計するだけの贅沢がない場合、予想される多数の集合問合せを予め計算し、これをテーブルに入れることができる。例えば、大型小売店が複数の店舗または複数の製品にわたる総売上げの合計を問い合わせることが多い場合、このような情報を特殊テーブルに記憶させることができる。このような方式の主要コストは、そのテーブルを更新の面に維持することである。(ディスクのスペースはもはや主要要素ではない。) その例では、店舗Sで品目Iを売り上げるごとに、Iの全製品売上げテーブルおよびSの全店舗売り上げテーブルを更新しなければならない。したがって、この方式は、問合せと問合せの間に更新が少ない場合は、価値がある。多い場合は価値がない。

【0010】

(i v) 最適化した外部キー結合

多次元テーブルの問合せは大部分が、中心の「事実テーブル」(例えば、売上げの詳細)と次元テーブルのセット(例えば、店舗の記述、製品の記述、顧客の記述)との間の結合を含む。これは、例えば、売上げテーブルの顧客識別子が、顧客記述テーブルのキーであるので、「外部キー結合」として知られる。(キーは、ある属性に属する値であり、したがってその属性にその属性を有するレコードは1つしかない。) この結合を加速する1つの方法は、事実テーブル・レコードと次元レコードとの間にリンケージを生成する。これは、以下のように3つの基本的方法で実行することができる。

(a) 事実テーブル・レコードの識別子と次元テーブル・レコードの識別子を保持する索引を作成する。

(b) 事実テーブル・レコードと次元テーブル行との間に双方向ポインタを作成する—これは「オブジェクト指向」のデータベースが実行することである。

(c) 事実テーブルの顧客レコード識別子を次元テーブルへのオフセットで置換する。

【0011】

選択肢(a)は、テーブルの物理編成の変化から最も影響を受けず、したがって更新が頻繁なシステムには最善である。何故なら、次元テーブルの変更を、そのテーブルの索引のみに反映できるからである。選択肢(b)は、物理編成に対して最も融通性が低い。次元テーブルの再編成が、事実テーブルの更新を含むからである。選択肢(c)は、事実テーブルを変更せずに、次元テーブルに対してその特定の物理的再編成を実行できる(例えば、ディスク上の位置を変更する)という点で、その2つの折衷である。結合最適化の例は、米国特許第5, 548, 754号、第5, 671, 403号、第5, 724, 568号、第5, 752, 017号、第5, 761, 657号および第5, 822, 747号で見ることができる。

【0012】

(v) 近似法

大抵の人は、戦略的な集合情報を取得するためにデータ・ウェアハウスを使用するので、多くはエラー限界を有する限り、高速近似法で満足する。この分野の典型的研究が、米国特許第5, 870, 752号で詳しく記述され、これは、エラー限界を与えながら、データ・ウェアハウスの集合結果を推定する方法を示す。基本的問題は、全テーブルのサンプリング、および次の集合の実行が、一般に働かないことである。例えば、キー上でRとSを結合したい場合、それぞれの1/10サンプルをとると、サンプルがランダムであれば、実際の結合のサイズの1/100であるサイズが与えられる。したがって、より賢くな

ければならない。その概念は、外部キー結合によって連結されるテーブル R, S, T, . . . の初期セットをとることである。例えば、R が事実テーブルで、その他が次元テーブルであるとする。R のサンプルをとり、次に R', S', T', . . . を与えるサンプルに基づいて、これらの外部キー結合を全て実行する。次に、問合せが R, S, T に関連し、その間の外部キー連結を含む場合、照会は、R', S', T' に関して非常に正確に実行することができる。エラーは、R' の幾つかの区画によって獲得された結果を考察し、その変化を調べることによって推定することができる。

【発明の開示】

【発明が解決しようとする課題】

【0013】

10

本発明の目的は、効率的な問合せ処理を可能にするデータベース管理システムを編成する代替方法を提供することである。

【課題を解決するための手段】

【0014】

本発明は、添付の請求項 1 から請求項 248 で述べるように、データベース・システムで情報を編成し、問合せを処理する方法を提案する。本発明は、さらに、このような方法を実施するように配置され、プログラムされた手段を備え、データの集まりから情報を管理するデータベース・システム、さらに、このような方法を実行するための命令を有するコンピュータ・プログラム製品を提案する。

【0015】

20

本発明の別の態様は、コンピュータ・システムで整数リストを扱う方法に関する。非制限的な方法で、本発明のこの態様は、RDBMS の分野に適用することができ、整数リストは、様々なテーブルのレコードの識別子を表すことができる。

【0016】

コンピュータ・システムでは、整数リストを、整数リストの明示的形態で、またはビットマップ・ベクトルの形態で同等に記憶し、扱えることがよく知られている。ビットマップ・ベクトルは 2 進成分を有し、各成分は、成分のランクに対応する整数がリストに属する (1) か、属さない (0) かを示す。ベクトルの次元は、少なくともリストの最大整数と等しくなければならない。

【0017】

30

ビットマップ表示が都合がよいのは、ベクトルの 2 進成分を通常のプロセッサの最も基本的な演算であるブール演算にかけることにより、コード化されたリストで多様な操作を実行できるからである。例えば、整数リストは、ブール AND 演算によって容易に交差したり、ブール OR 演算によってマージしたり、ブール NOT 演算によって補足したりする。

【0018】

リストの整数が潜在的に大きい場合は、ビットマップ・ベクトルの次元が大きくなり、したがってその形式でリストに記憶するために必要なメモリ・スペースが問題となる。リストが、大きい範囲の整数で充填されることが稀である場合、明示的整数フォーマットは、はるかにコンパクトになる。すなわち、範囲 $[0, 2^{32}]$ の整数 K 個のリストは、ビットマップ・フォーマットに $K \times 32$ ビットすなわち、 $2^{32} \times 43$ 億ビットが必要である。

40

【0019】

ビットマップ表示のこの限界を克服するため、ビットマップ圧縮法が提案されている。この方法は、成分の値が定数であるベクトルの領域を、この領域の境界のみを符号化するように位置決めすることにある。残りの領域は、ビットマップ・セグメントとしてコード化することができる。非常に大きい定数の領域が見られる場合に、相当の利得が獲得される。このようなビットマップ圧縮法の例は、米国特許第 5,363,098 号および第 5,907,297 号で開示されているようなものである。

【0020】

このタイプのビットマップ圧縮は、符号化した整数リストの記憶を最適化するが、その扱

50

いは最適化しない。ビットマップで基本的ブール演算を実行する場合、重複したビットマップ・セグメントを検出するために、複数の比較が必要である（米国特許第6,141,656号参照）。これは計算にとって効率的ではない。また、定数領域およびビットマップ・セグメントのコード化データを、ハード・ドライブ（すなわち、RAMではない）などのメモリ装置に記憶する場合は、通常、多数のディスク読み取り演算が必要であり、これは処理速度にとって好ましくない。

【0021】

本発明のこの態様の目的は、整数リストを符号化および／または結合する代替方法を提供することであり、これにより潜在的に大きい次元になるリストを効率的に扱うことができる。

10

【0022】

したがって、本発明は、添付の請求項249から291で述べるように、コンピュータ・システムで整数リストを符号化し、結合する方法を提案する。本発明は、さらに、このような方法を実施するように配置され、プログラムされた手段を備えるコンピュータ・システム、さらにこのような方法を実行する命令を有するコンピュータ・プログラム製品を提案する。

【発明を実施するための最良の形態】

【0023】

仮想データ・グラフ

図1～図3は、リレーショナル・データベース管理プログラムを実行するよう構成されたプロセッサと結合したコンピュータ・メモリに記憶することができるデータの集まりを示す。この例は、以下の説明では、適宜、本発明の原理および実施形態を例示するよう言及される。

20

【0024】

図1～図3は、データベース・システム内の従来タイプのデータ編成を示す。図示のシステムは、顧客のための保険を管理する仮想保険会社に関連するデータを扱う。データは、それぞれ図1～図3に示すように、顧客、保険および事故に関する3つのテーブルで編成される。

【0025】

論理的見地から、各データ・テーブルは、2次元マトリクスで構成され、行はテーブルの個々のレコードに対応し、列はレコードの個々のデータ属性またはデータベースの構造的特徴に対応する（后者のタイプの列は、通常、構内レコード識別キーまたは目標テーブル内でレコードを指定する外部キーを含む）。

30

【0026】

しかし、大きいデータベースでは、メモリ媒体、例えば、磁気ディスクへの実際のデータ記憶は、往々にして他の方法で実行されることが理解される。すなわち、各行は通常、メモリ・アドレスを有し、ここに対応する属性値またはキーが列の順序で記憶され、符号化された文字「\」などの所定の記号で分離される。

【0027】

提案するデータ構造の説明を容易にするために与えられる簡略化した例では、テーブルはほどほどのサイズである。実際には、通常、これよりテーブルおよびテーブル当たりの属性（列）が多い（それにもかかわらず、1つまたは複数のテーブルが1つの列を有することもある）。さらに、データ・テーブルは、通常、用途に応じて最高数千または数百万行など、はるかに多くのレコードを含む。

40

【0028】

その例のデータベースでは、属性7個のグループが、3つのデータ・テーブルに対応する3つのサブグループに分布する。各属性は、そのサブグループに対応するデータ・テーブルに列を有する。顧客データ・テーブル（図1）は3つの属性、すなわち、顧客名、誕生日および性別を有する。図2の保険データ・テーブルは2つの属性、すなわち、保険のタイプ（「自動車」または「住宅」）および保険の有効期日、および顧客テーブルへの連結

50

列を有する。図 3 の事故データ・テーブルは 2 つの属性、すなわち、事故日および発生した損害賠償の任意の通貨での金額、および保険テーブルへの連結列を有する。

【0029】

任意のデータ・テーブルにおいて、各レコード／行は一意の識別子を有し、これは行 ID と呼ばれる。この識別子は、レコードが記憶されるメモリ・アドレスに対応し、通常は変換テーブルを通す。これは、一意の行識別のために、データ・テーブルの列の識別キーとして記憶することができるが、これは強制ではない。本明細書の例では、行 ID は、データ・テーブルごとにゼロから開始する整数索引であり、テーブルの列には明示的に記憶されない。

【0030】

図 2 および図 3 の最終列で示すように、テーブルの幾つかを相互に連結する。2 つのテーブルを直接連結するのは、その一方（ソース・テーブル）が、レコードを他方（目標テーブル）に指定する外部キーを含むために設けた連結列を有する場合である。

【0031】

これらの外部キーは、これ以降はリンクと呼び、リレーショナル・データベース・システムで扱うデータの階層および編成を反映する。本明細書の例では、保険会社が扱う各事故は、同社が管理する特定の保険、すなわち、図 3 の保険リンクに関連付けられる。各保険は、同社の特定の顧客用であり、したがって図 2 の顧客リンク用である。一部のリンクは任意選択でよいことを理解されたい。例えば、第三者が関与する事故もあり、第三者用に別個のテーブルがある場合、事故テーブルの各レコードは、第三者テーブルへのリンクを有する。

【0032】

各リンクは、通常、目標データ・テーブルの行 ID で構成される。例えば、図 3 の事故テーブルで行 ID = 0 として記憶され、1998 年 10 月 3 日に発生し、1,000 の金額となる事故は、図 2 の保険テーブルで行 ID = 1 として記憶された保険を指す保険リンクを有する。すなわち、顧客が、図 1 の顧客テーブルの行 ID = 1 で 1998 年 9 月 9 日に加入した自動車保険、すなわち、1976 年生まれの男性 Andre に関連する。目標テーブルが他の形式のレコード識別キー、例えば、複合キーを有する場合、リンクは、このようなキーによって識別される目標レコードも指定する。

【0033】

リンクの構造は、幾つかの規則に従う。特に、リンクされたデータ・テーブルは、図 4 に示した階層木編成のような有向非周期グラフ構造を有する。根テーブルは、図 3 の事故テーブルのように、他にその行を指すリンクを有するデータ・テーブルがないデータ・テーブルと定義される。すなわち、根テーブルは、目標テーブルを構成しない。同様に、葉テーブルは、図 1 の顧客テーブルのように、リンク列がないデータ・テーブルと定義される。すなわち、葉テーブルは、ソース・テーブルを構成しない。図 4 は根テーブルを 1 つしか示さないが、テーブルの木構造は複数の根を有することができる。

【0034】

特定のケースでは、関連したデータ・テーブルのグループが循環リンクを呈することもある（例えば、顧客テーブルは、各顧客が遭遇した最初、または最近の事故を示すため、事故データ・テーブルへのリンク列を有することがある）。このようなケースでは、データ・テーブルの木編成は、最初に循環の 1 つのリンクを取り消すことによって復元される。どのリンクを取り消すかは、データベースのセマンティクスによって指示される（上記の例では、当然、顧客テーブルから事故テーブルへのリンクが取り消される）。

【0035】

データ・テーブル内で、根テーブルから葉テーブルへの経路が定義される。根テーブルから葉テーブルへの各経路は、葉テーブルを指す根テーブルのリンク列によって定義されるか、1 つまたは幾つかの中間テーブルを介して一続きのリンク列によって定義される。

【0036】

図 4 では、複数の分岐がある木構造を示すため、2 つの葉テーブル（点線）が追加されて

10

20

30

40

50

いる（図1～図3の簡略化した例は、実線で示した単経路の木を提供する）。追加された葉テーブルは、前述したような第三者テーブル、および保険テーブルからの目標テーブルである仲介者テーブルであり、保険を商品化する仲介者に関するデータを含む。

【0037】

相互に連結されたデータ・テーブル・レコードを、同様の木表示（図5～図7）で見ることができる。図5のレコード木は、事故#6が、顧客#2（A r i a n e）が仲介者#Yを通して加入し、第三者#Xを巻き込んだ保険#0（自動車）に関連したことを示す。実線は、図2および図3のデータ・テーブルからの個々のリンクを表す。

【0038】

図6のレコード木は、さらに、図3の最終列から明白なように、保険#2（顧客#4（M a x）が自宅について加入した）で事故が発生しなかったという理由で、保険テーブルの行ID=2へのリンクを有する事故テーブルに追加できる空レコードを示す。

【0039】

空、すなわち、ダミー・レコードは、データが存在しないことを示す。その属性値は全てデフォルト値（ゼロ）であり、「値なし」を意味する。この方式にこのようなダミー・レコードを挿入する目的は、どのデータ・テーブルにあるどの有効レコードも、根テーブルのレコード（図4）に由来する少なくとも1つのレコード木に属することを確認することである。

【0040】

空レコードは、ソース・テーブルの少なくとも1つのリンク列の目標テーブルである各データ・テーブルにも存在することができる。ソース・テーブルの行に、対応するリンク列の外部キーがない場合、その行を含むレコード木は、前記目標テーブル位置を空にすることで完成する。この状況は、図6に示した例の仲介者テーブルで発生している。それを示すため、ソース・テーブルのリンク列にデフォルト値（例えば、-1）を書き込むことができ、これによって目標テーブルに明示的に空レコードが存在する。

【0041】

空レコードは、適宜、前記経路の葉テーブルから根テーブルへと、すなわち、図4の下方向にデータ・テーブルの各単経路を走査するプロセスに挿入される。経路の走査中に1つのソース／目標テーブル対を検査する場合は、ソース・テーブルの関連するリンク列で発生しない目標テーブルの行ID値が最初に、次にリストの各ミッシング行ID値がリストされ、前記ミッシング行IDが前記リンク列にある状態で、ソース・テーブルに新しい空レコードが生成される。

【0042】

幾つかのリンク列を有するデータ・テーブルに空レコードをこのように挿入すると、空レコードは、走査中の経路に関する列以外のどのリンク列でもデフォルト値（-1）を受け取り、対応するリンクが目標テーブルの空レコード宛てであることを示す。この状況は、図6に示した例の第三者テーブルで発生している。

【0043】

データ・テーブル木を葉から根へと走査することが重要である。そうしないと、目標テーブルの他の空レコードへのリンクを含む空レコードを見過ごすことがある。例が図7に図示され、これは保険が（もう）ない顧客#0（O s c a r）に関するレコード木を示す。事故テーブルは、保険テーブルの別の空レコードを指す空レコードを含み、保険テーブルは、顧客#0を指す。経路を上方向に走査すると、レコード木の根が根（事故）テーブルにない。

【0044】

図1～図3に示すような従来のデータベース編成では、メモリの使用を最適化するためにリンク・キーを設ける。それを示すため、図8に示したフラットファイルを参照することができ、これは図1～図3の3つのデータ・テーブルと全く同じ情報内容を有する（結局、第三者および仲介者テーブルは無視される）。

【0045】

フラットファイルは、データ・テーブルの属性（列）ごとに列を有する。図4のデータ・テーブル木構造で構築できる完全な各レコード木で、フラットファイルは、関連の列に、前記木の全レコードの属性値を含む行を有する。フラットファイルの行は、本明細書ではデータ・グラフと呼ぶ。各データ・グラフは、図8の左手部分に示すフラットファイル行IDで識別される。図5～図7のレコード木は、それぞれ行ID6、9および11におけるデータ・グラフのコンパクトな表示である。

【0046】

フラットファイルの表示を文献と呼ぶことがあるが、大きいサイズのデータベースにとっては実際上ほとんど関係がない。データ記憶に過度の冗長性を必要とするからである。

【0047】

例えば、本明細書の小さいサイズのケースでは、Andreの生年および性別、さらにその自動車保険の詳細が、フラットファイルに3回（行ID0、3および8）書き込まれるが、記憶が図1～図3のようなデータ・テーブルの形式である場合は、リンク値とともに1回しか書き込まれない。現実的なサイズのデータベースでは、このような冗長性は許容できない。

【0048】

本発明によるデータベース・システムは、フラットファイルの概念を使用する。しかし、図8に示すようなフラットファイルの記憶を必要とせず、したがって「仮想フラットファイル」も概念は「仮想データ・グラフ」（VDG）を含む。「仮想」という用語は、フラットファイルまたはデータ・グラフをメモリに明示的に保持する必要はないが、方法を実行時にそのデータ構造を参考として使用することを指す。

【0049】

本発明の特定の実施形態では、フラットファイルは、図9に示すようなリンク・テーブルに還元される。リンク・テーブルの各行は、フラットファイルの個々の行、すなわち、図5～図7に示すようなレコード木に対応する。

【0050】

リンク・テーブルの列は、それぞれ、図1～図3のデータ・テーブルに対応する。すなわち、リンク・テーブルの各列は、対応する（目標）データ・テーブルに割り当てられた属性のサブグループである属性サブグループに関連付けられる。リンク・テーブルの各列は、対応する目標データ・テーブルのレコードを指定するリンク値（行ID）を含む。

【0051】

任意のデータ・グラフに対応するリンク・テーブルの行は、前記データ・グラフを表すレコード木に空レコードを有する全てのデータ・テーブルに対応する列に、デフォルト値（-1）を含む。

【0052】

リンク・テーブルの1つの行にデータ・テーブルの行IDが見られたら、データ・テーブルからリンクされたデータ、すなわち、データ・グラフまたはその一部を検索することができる。リンクは全てリンク・テーブルに表示される。図9のリンク・テーブルの列で記憶された行IDを、図1～図3の個々のデータ・テーブルで識別された行に記憶された属性値と置換すると、図8のフラットファイルが回復される。

【0053】

提案するシステムは、さらに、それぞれがデータ・テーブルの1つの各列と、すなわち、属性の1つの関連付けられたワード・シソーラス（図10A～図10G）を使用する。

【0054】

好ましい実施形態では、データベース・システムで使用する属性ごとに1つのワード・シソーラスがある。しかし、問合せ基準で、全く使用しない、またはほとんど使用しない属性があると分かっている場合は、このような属性について、シソーラスなしで済ますことが可能である。

【0055】

データ・テーブルの1つの列に関連付けられた各ワード・シソーラスは、その列に見られ

10

20

30

40

50

る各属性値の項目を有する。このような属性値を、本明細書では「ワード」と呼ぶ。ワードは、関連するデータ・テーブルの列で少なくとも1回発生する限り、シソーラスに1つの項目、そしてわずか1つの項目を有する。

【0056】

各シソーラスの項目は、属性値に基づいて記憶される。したがって、属性のカテゴリごとに順番関係が定義される。シソーラス・ファイルの属性値フィールドを定義し、次元決定する場合は、これに注意が必要である。

【0057】

通常、ワードはASCIIフォーマットであり、そのカテゴリは、列ごとに「整数」、「実数」および「文字ストリング」のカテゴリから選択される。文字ストリングは、通常の辞書式順序に従ってソートされる。日付フィールドは、y y y y (m m) (d d) (図10B、図10Eおよび図10F)のような文字ストリングとして宣言することが好ましく、ここで、y y y yは年、m mは月(任意選択)、d dは日(任意選択)を表す。このようにソートするシソーラスは、日付を入力順にする。属性カテゴリが「整数」である場合は、整数のデータ値間に自然の順序関係を提供するため、数字を右揃えする。属性カテゴリが「実数」である場合は、その経路全体に従って数字を揃え、列で最長の少数部分を有する値と同じ桁数が右側にある。

【0058】

空値が、記憶された各シソーラスの一方端(例えば、冒頭)にある。

【0059】

データ・テーブルTの列C(T)に関連付けられたシソーラスのワードWの各項目E(W)が、C(T)に対応する列に属性値Wを有するフラットファイルの各行を識別するための情報を含む。フラットファイルをリンク・テーブルの形式で仮想的に記憶する場合は、データ・テーブルTに対応する列内に、列C(T)に値Wを有する行を指すリンクがあるリンク・テーブルの各列を識別するために、項目E(W)に含まれた情報を使用する。

【0060】

すなわち、シソーラスの項目E(W)の内容が列C(T)と関連する状態で、その対応する属性が値Wを有する全データ・グラフを検索することができる。

【0061】

このような内容は、(仮想)フラットファイルの行を指す行IDリスト、すなわち、データ・グラフ識別子リストを表す。このようなリストは、(図10A～図10Cのように)特に一部のシソーラスの空値で、空白になることがある。

【0062】

シソーラスのデータ・グラフ識別子リストの2つの代替表示を、図1～図3の7つの属性列について、図10A～図10Gに示す。最初のものは、明示的整数リストの形式である。

【0063】

第2の(等価)表示は、長さが、仮想フラットファイルの行数、すなわち、データ・グラフの数に等しい(またはこれより大きい)ビットマップ・ベクトルの形式である。ビットマップ・ベクトルの位置iのビット($i \geq 0$)は、その整数iが、ビットマップ・ベクトルで表される行IDリストに属するか(1)属さないか(0)を示す。本明細書の簡略化した例では、フラットファイルは12の行を有し、したがってビットマップ・ベクトルは次元12でよい。

【0064】

上述したデータ構造は、仮想フラットファイル、および記憶され、仮想フラットファイルの行を指すシソーラス・ファイルを備え、本明細書ではVDG構造と呼ぶ。

【0065】

VDG構造は、データベースで問合せを効率的に処理するための強力なツールを提供する。

【0066】

仮想フラットファイルは、全シソーラスの項目の統一された代数枠組を定義する基準テーブルである。問合せの基準に関連するシソーラスを参照して検査し、問合せ基準に一致するデータ・グラフがあれば、それを全て表すフラットファイル行 I D リスト（またはビットマップ・ベクトル）を獲得する。次に、所望に応じて、結果の表示の属性値を検索するため、その行 I D リストで指されたリンク・テーブル行にアクセスし、列の一部または全部に現れるリンクを読み出すことにより、結果を配布することができる。

【0067】

シソーラスに関する処理は、主に、これが指す行 I D リストで実行する論理演算である。整数リストとして表される場合、このような演算は、基本マージ、交差および／または相補演算に還元することができ、これはそれぞれ、ビットマップ表示のブール O R、A N D、N O T 演算に対応する。 10

【0068】

V D G 構造は、データベースの内容にアクセスするための効率的なツールも提供し、これはデータ・テーブルへのアクセスを必要としない。このツールは、S O R T、C O U N T、D I S T I N C T、O R D E R B Y、G R O U P B Yなどの文節のような特殊な結果表示機能を有し、問合せに一致するレコードのデータ値で任意のタイプの計算を実行するための問合せに非常に適している。

【0069】

例 1：

例示として、以下の問合せを考察されたい。すなわち、900より高い損害賠償が発生する全自動車事故の顧客名および事故日を検索し、顧客名に従って結果をグループ分けする。問合せは、以下のように処理することができる。最初に、900より高い金額に関する事故金額シソーラス項目で識別されたフラットファイル行 I D リスト（図 10 G の最後の 5 行）をマージすると、リスト {0, 1, 3, 5, 6, 7}（またはビットに関するブール O R によって獲得されたビットマップ・ベクトル 110101110000）が生成される。次に、そのリストと、「自動車」の値に関する保険タイプ・シソーラス項目で識別された行 I D リスト（図 10 D の第 2 行）との交差を決定する。結果リスト {0, 3, 5, 6}（またはビットに関するブール A N D で獲得されたビットマップ・ベクトル 100101110000）は、問合せの基準を満足するデータ・グラフを指定する。最後に、顧客名シソーラス（図 10 A）の項目を順次読み出し、結果リストと顧客名シソーラス項目で識別された行 I D リストとの間に空白でない交差がある場合、その交差でその行 I D を有するリンク・テーブル行を読み出し、所望の属性値を検索する。本明細書のケースでは、出力は、A n d r e [事故日 1998 年 10 月 03 日 (#0) および 1999 年 06 月 12 日 (#3)]、A r i a n e [事故日 1999 年 12 月 09 日 (#6)] および L a u r e [事故日 1999 年 12 月 08 日 (#5)] となる。 20 30 40

【0070】

上記のタイプの処理は、任意の種類 of 問合せに適用することができる。応答は、フラットファイルの枠組に明示的に言及するソートされたシソーラスのみを参照することによって準備される。出力フラットファイル行 I D リストが獲得されたら、問題となるデータを検索するため、リンク・テーブルまたはシソーラスを使用することができる。 40

【0071】

例 2：

V D G 方式の優れた性能をさらに例示するため、当人の 35 回目の誕生日となる常用年の開始前に自動車事故に遭遇した顧客を識別するという問合せについて考察してみる。従来の典型的なシステムでは、図 3 の事故データ・テーブルの全レコードを読み出して、日付の属性および保険リンク値を獲得しなければならない。事故レコードごとに、保険リンク列で見られる行 I D で保険データ・テーブルを読み出し、保険タイプの属性および顧客リンク値を獲得して、次に保険タイプが「自動車」である場合は、顧客リンク列にある行 I D でさらに顧客データ・テーブルにアクセスし、生年属性値を獲得する必要がある。後者の値を事故テーブルで以前に獲得した日付属性値と比較して、問合せの基準を満たすか決 50

定する。

【0072】

事前に問合せ基準で参照した属性に基づいてデータ・テーブルをソートしている場合、このような従来の処理は、ディスク・アクセス数を制限することによって加速することができる。それには、レコードを追加、削除または更新するたびにデータ・テーブルのソートが必要であり、これはデータの集まりが大きい場合に实际的ではない。さらに、データ・テーブル・ソート規則で指定された特定のケースでしか可能ではない。

【0073】

例えば、生年および保険タイプの属性に基づいて顧客および保険テーブルを個々にソートした場合、適切な順序で、コンピュータのキャッシュ・メモリの助けを受けてデータ・レコードにアクセスすることにより、先行する要求を、それほど過酷でない時間で処理することができる。しかし、このコツは、他の同様の問合せには当てはまらない（例えば、超過額を含むために、保険テーブルに追加の列があるとすると、損害賠償額が超過額の10倍より大きい事故全部の識別で、同じ問題が生じる）。 10

【0074】

V D G方式では、上記の例示としての問合せを、非常に効率的な方法で扱うことができる。顧客の生年シソーラス（図10B）および事故日シソーラス（図10G）によって、コンピュータは、日付の基準を満足する{顧客の生年、事故日}ワード対、すなわち、顧客の生年+35の開始より前の事故日を識別する。これは、事故を起こしたのが顧客か否かを心配することなく実行される。このような識別は、問合せで言及されるような属性は全て、ソートされたシソーラスを有するので、可能ないずれの属性対でも比較的容易である。識別されたワード対ごとに、シソーラス項目の2つのフラットファイル行IDリストの交差が獲得される。その結果生じた整数リストをマージする。次に、コンピュータは、保険タイプ・シソーラスの「自動車」の値に関連する項目の行IDリスト（図10Dの第2行）と、マージから結果するリスト{0, 1, 3, 5, 6, 8, 10}とを交差させる。結果のリスト{0, 3, 5, 6, 8}は、リンク・テーブルの1組の一致列を指定し、そこから、一致行およびリンク・テーブルの顧客列に表示される行IDを有する顧客テーブル・レコードにアクセスすることにより、関連する顧客名（Andre-3回一、Lau 20
reおよびArianne）が容易に検索される。

【0075】

問合せを処理する場合は、リンク・テーブルが、問題のデータを検索する手段として簡単に使用されることに留意されたい。この検索機能を遂行する様々な方法を想定することができる。 30

【0076】

1つの方法は、元のデータ・テーブル（図1～図3）をメモリに保持する。しかし、その内容が既にリンク・テーブルに存在するので、リンク列がこのデータ・テーブルから削除されていることがあることに留意されたい。

【0077】

可能な属性値が全て、対応するシソーラスに記憶されているという観察から、別の方法は、シソーラスへのリンク・テーブル・ポインタに記憶することである。後者の方法は、データ・テーブル列内で値が頻繁に発生しても、属性値を1回しか書かず済むので、必要なディスク・スペースが減少する。これによって、元のデータ・テーブルを使用する必要なく、任意のフラットファイル行で発生する属性値を迅速に検索することができる。 40

【0078】

特定の属性では、明示的属性値をリンク・テーブルに、すなわち、フラットファイルと同様に記憶すると魅力的なことがある。特に、これは、その値が非常に分散し、問合せ応答の出力属性（例えば、金額）の中で往々にして要求される数値フィールド（通常は文字ストリングよりサイズが小さい）で魅力的である。これらの値が、リンク・テーブルに明示的に書き込まれている場合は、必要なディスク・スペースの費用上昇をほどほどにして、出力データの取り出しに必要なディスク・アクセスに、大きい利得があることがある。 50

【0079】

以上の説明では、リンク・テーブルは、一種のフラットファイルの骨組みであり、これは、記憶したシソーラスによって問合せに一致するフラットファイル行IDのリストが決定されていたら、データ検索を容易にするために記憶される。

【0080】

それにもかかわらず、フラットファイル構造を反映するリンク・テーブルまたはテーブルの形式を記憶することが、極めて必要である。有利な実施形態では、データ・グラフ（または、その結果表示のために要求される部分）は、シソーラス・ファイルのみから回復することができる。それを例示するため、再び例2について考える。シソーラス・ファイルに関連して問合せ基準を処理することにより、一致する仮想フラットファイル行の結果リスト{0, 3, 5, 6, 8}が獲得されたら、顧客名シソーラスを走査し、ワード（顧客名）ごとに、シソーラスで表されたフラットファイル行IDを結果リストと交差させることが可能である。交差が空白でない場合、そのワードは出力に含まれる。これは、交差リストを伴い、それによってユーザは、関連するデータ・グラフからさらなる情報を迅速に獲得することができる。この方法は、シソーラス・ファイルしか記憶する必要がないので、必要なメモリ・スペースが最小になる。

【0081】

リンク・テーブルが記憶されている場合でも、特定の問合せでは、リンク・テーブルを通してよりも、上記で示したようにシソーラスを走査して属性値を検索すると有利なことがある。これは、特に問合せプロセッサとデータ・テーブル間に比較的遅いインタフェース、例えば、ODBCインタフェース（「オープン・データベース接続性」）がある場合に、データ値に関する計算を実行する時に発生する。

【0082】

VDG方式のもう1つの利点は、元の形式でデータ・テーブルと共存できる問合せ処理エンジンを提供することである。次に、元のデータ・テーブルの対応する変化に応答して、シソーラス項目の変更を実行する。これは、従来のテーブルの形式でデータを維持することが重要であることが分かったユーザにとっては魅力的な機能である。何故なら、新しいシステムに過度に依存したくない、または他の用途用の従来のインタフェースを通してテーブルにアクセスする必要があるからである。

【0083】

マクロワード

上述したVDGは、マクロワード・シソーラスとも呼ばれる接頭シソーラスで終了すると有利である。

【0084】

上述したワード・シソーラスと同様、各マクロワード・シソーラスは1つの属性、すなわち、1データ・テーブルの1つの列に関連付けられる。また、接頭語長（または打ち切り長）パラメータを有する。

【0085】

マクロワード・シソーラスの各項目は、ある範囲の属性値に関係し、前記属性に対応する列に、前記範囲に入る属性値を有するフラットファイル行を全て識別するためのデータを含むか、それを指す。マクロワード・シソーラスの項目に対応する範囲は、そのシソーラス割り当てられた接頭語長を有する接頭語値に対応する。すなわち、このような接頭語値で開始するワードは、マクロワードのそれに含まれるそのフラットファイル行IDリストを有する。接頭語長がPと表記される場合、マクロワード $C_1 C_2 \dots C_P$ は、P個の文字または数字 $C_1 C_2 \dots C_P$ で始まる属性の全値のセットである。接頭語長が、値フィールドの文字または数字の数である（すなわち、切り捨て長さがゼロである）限定的ケースは、前述したワード・シソーラスである。

【0086】

すなわち、マクロワード・シソーラス項目は、マクロワードに含まれた個々のワードに対応するワード・シソーラスの項目で識別されたフラットファイル行IDリストのマージ（

またはビットマップ・ベクトル間の論理OR)に対応するフラットファイル行リスト(またはビットマップ・ベクトル)を識別する。

【0087】

したがって、属性ATに関連付けられる各シソーラス(ワードまたはマクロワード)は、関連するデータ・テーブル・レコードの属性ATに割り当てることができるワードのセットのサブセットへの区分に関連して定義することができる。これは、属性ATに割り当てられた少なくとも1つのワードを含むサブセットごとに個々の項目を有し、この項目は、属性ATに割り当てられたサブセットのワードを有するフラットファイル行のIDを含むフラットファイル行IDリストに関連付けられる。マクロワード・シソーラスのケースでは、区分は、各サブセットが、共通接頭語で開始するワードで構成されるような区分である。ワード・シソーラスのケースでは、区分は、各サブセットがワード1つだけで構成されるような区分である。 10

【0088】

一例として、図10Hは、3文字の切り捨て長についての事故金額マクロワード・シソーラスを示す。空項目を反復する必要はない。これは既にワード・シソーラスにある。このようなマクロワード・シソーラスは、ディスクのアクセスおよびフラットファイル行IDリストのマージの面で非常に経済的である。例えば、1,000から1,999の間の金額を有する事故に関する情報を獲得するため、フラットファイル行ID{0,3,6}の関連リストを獲得するには、図10Hのマクロワード・シソーラスにアクセスすれば十分であり、図10Gの切り捨てがない事故金額では、2つのシソーラス・アクセスおよび1つのマージ演算が必要になる。利得は、大きいデータベースおよび高い濃度の属性、すなわち、可能な属性値が多い場合に非常に大きくなることがある。 20

【0089】

接頭語または切り捨て長に基づくマクロワード・シソーラスは、範囲に基づく問合せ基準の処理に、大きい融通性を提供する。任意の属性で、様々な問合せの処理速度を最適化するために異なる接頭長を有する幾つかのマクロワード・シソーラスを提供することが可能である。

【0090】

通常、日付の属性は、(日)ワード・シソーラスに加えて、年マクロワード・シソーラス(接頭語長=4)および月シソーラス(接頭語長=6)を有することができる。他の任意の種類属性(数字またはテキスト)も、都合がよいマクロワード・シソーラス階層になることがある。 30

【0091】

V D G 圧縮

これまでに説明したようなV D G方式では、シソーラス・ファイルで要求されるメモリ・スペースが最適化されていない。

【0092】

行IDが、通常は32ビットでコード化される整数である状態で、ワードが図8のフラットファイルの属性列でn回発生する場合、そのフラットファイル行IDリストを明示的に符号化するには、 $n \times 32$ ビットが必要である。フラットファイルが N_{max} 行(例えば、数百万行)を有する場合、nの値が何であっても、ビットマップ表示の各項目に N_{max} のビットが必要である。 40

【0093】

概して、日付または金額属性(図10E~図10G)などの密度が高い属性では、フラットファイル行IDリストが満杯になることは稀であり、したがってメモリの要件の面で明示的整数リスト表示は満足できるものであり、ビットマップ表示は、大きいフラットファイルには過度に重いことがある。本明細書の例にある顧客の性別または保険タイプ属性(図10C~図10D)のように、密度が低い属性もあり、これによってビットマップ表示が非常に適しているが、整数リスト表示は好ましくない。

【0094】

各シソーラスに、必要なメモリ・スペースを減少させるために最も適切と考えられる表示を適用することが可能である。しかし、それには、属性値を分散させる方法について先験的な知識が必要である。この点で多くの属性が不明瞭であり、任意の属性に関する様々なマクロワード・サイズでは、最適化も困難なことがある。

【0095】

当業者であれば周知のビットマップ圧縮方法（例えば、米国特許第5,363,098号または第5,907,297号）を使用してもよい。問題は、これらの方法が基本的に、処理速度ではなく、記憶量を最適化するために設計されたものであることである。VDGの状況では、メモリ・スペース減少の利点は、問合せ処理時の複数の圧縮および／または復元作業による応答時間の延長という欠点によって相殺されることがある。これに対して、処理速度は可能な限り上げることが望まれる。

【0096】

VDG方式の好ましい実施では、シソーラスでのフラットファイル行IDリストの圧縮は、フラットファイルの行ID全部にわたる範囲を所定のパターンに従ってサブセットに分割することによって実行される。次に、シソーラス項目の各フラットファイル行IDリストは、リストの少なくとも1つの行IDを含む範囲の各サブセットをパターン状に位置づけるためのデータ、およびこのように位置づけられたサブセット内の行IDリストの各整数の位置を表すデータで符号化される。

【0097】

行IDの範囲 $[0, N_{max}]$ は、そのファイルにある行の数以上になるよう選択される。 「所定のパターン」は、「サブセット」を、前記範囲内に同じ長さ $D1$ を有する連続的インターバル $[0, D1-1]$ 、 $[D1, 2 \times D1-1]$ などと都合良く定義する。

【0098】

コード化データは、ユークリッドの除法で非常に簡単に生成することができる。任意の正の数 x および y について、 x に等しいまたはそのすぐ下の整数を

【0099】

【数1】

$$\lfloor x \rfloor$$

30

x と等しいまたはそのすぐ上の整数を

【0100】

【数2】

$$\lceil x \rceil$$

および

【0101】

【数3】

$$x \bmod y = x - \lfloor x/y \rfloor$$

40

と表記する。 $D1$ でのユークリッド除法を、入力リストの行IDの N ごとに実行する。商

【0102】

【数4】

$$Q1 = \lfloor N/D1 \rfloor$$

は、パターン内の対応するインターバルのランク（ $Q1 \geq 0$ ）を示し、剰余 $R1 = N \bmod D1$ は、インターバル内の行IDの位置を表す（ $0 \leq R1 < D1$ ）。復号も非常に簡単である。すなわち、コード化したリストの項目の符号化データ $Q1$ および $R1$ から、行IDは $N = Q1 \times D1 + R1$ である。

50

【0103】

インターバル長は2の整数累乗であり、したがってユークリッド除法は、簡単なビット・シフト演算で実行される。典型的な長さは $D1 = 2^7 = 128$ である。

【0104】

符号化方法は、ビットマップ表現に言及することにより、同じ意味で表現することができる。各ビットマップ・ベクトルをビットマップ・セグメント（または、より迂遠なパターンが好ましい場合は、他のタイプのビット・グループ）に分割し、少なくとも1つの「1」を含むセグメントごとに、コード化データは、ランク（= $Q1$ ）およびセグメントの内容を含む。ゼロ・セグメントは全て廃棄される。

【0105】

図11A、図11Gおよび図11Hは、図10Aおよび図10Gの顧客名および事故金額ワード・シソーラス、および図10Hの事故金額マクロワード・シソーラスを、 $D1 = 3$ （簡潔にするため、本明細書で使用する $D1$ の非典型的な値）にした他の表現である。第2列は、図10A、図10Gおよび図10Hそれぞれの最終列からコピーし、ビットマップ・ベクトルの区分を強調表示するブランクがある。第3列は、符号化の結果生成されるランクのリスト（= ユークリッド商 $Q1$ ）を示し、第4列は、対応する非ゼロ・ビットマップ・セグメント（各剰余 $R1$ の位置に1を有する）を示す。

【0106】

シソーラス項目ごとに、ランク $Q1$ は、

【0107】

【数5】

$$N1_{\max} = \lceil N_{\max}/D1 \rceil$$

で、範囲 $[0, N1_{\max}]$ に含まれる整数リストを形成することが観察される。

【0108】

本発明の好ましい実施形態によると、同様のタイプの符号化をこれらのランク・リストに適用することができる。符号化プロセスは、同じ符号化パターンまたは異なる符号化パターンで、数回反復することができる。特に、インターバル長は、反復することに変更することができる。これは2の整数累乗であることが好ましい。

【0109】

インターバル長 $D1$ で最初の反復にて獲得されるランクおよびビットマップ・セグメントは、層1（または $L1$ ）ランクおよび層1セグメントと呼ばれる（図11A、図11Gおよび図11H）。 $D2$ と表記されるインターバル長で第2の反復にて獲得されたものは、層2（または $L2$ ）ランクおよび層2セグメントと呼ばれ（図12A、図12Gおよび図12H）、以下同様である。

【0110】

以下では、 n は $1 \leq k \leq n$ で k と番号付けされた符号化層の数を意味し、層 k は除数パラメータ Dk を有し、積

【0111】

【数6】

$$\Delta k = \prod_{k'=1}^{k-1} Dk'$$

は、層 k ビットマップ・セグメント（ $\Delta 1 = 1$ ）の1ビットに含まれるフラットファイル行 ID の数である。

【0112】

図12A、図12Gおよび図12Hで示した簡略化したケースでは、 $n = 2$ であり、第2の符号化層は $D2 = 2$ を使用する。「 $L1$ ビットマップ」とラベルされた列は、層1ラン

10

20

30

40

50

ク・リストのビットマップ表現であり、最後の2列に示された層2データを導くさらなるビットマップ区分を強調表示するブランクがある。

【0113】

層1および層2コード化データが図13A、図13Gおよび図13Hに図示され、これはフラットファイル行リストの情報を記憶するのに可能な方法を示す。層1ランク・リストは、層2データで完全に定義されているので、そのリストの記憶は厳密には必要ないことに留意されたい。しかし、このような記憶は、本発明の特定の実施形態では、問合せ処理を多少簡略化することをさらに理解されたい。

【0114】

ワードおよびマクロワード・シソーラスの任意の1つに同じ種類の符号化を使用することが10
できる。しかし、その一部が従来タイプの行IRリスト記憶（明示的整数リストまたはビットマップ・ベクトル）、すなわち、 $n=0$ を保持することも可能である。特に、明示的整数リスト表現は、それでも希薄に分散したシソーラスによく適合している。

【0115】

図14～図16は、図13A、図13Gおよび図13Hのシソーラスに含まれる情報を記憶する可能な別の方法を示す。符号化した層ごとに、シソーラス項目は、ランク・ファイルおよびビットマップ・セグメントを含むデータ・コンテナ（層1については図15、層2については図16）にあるレコードの個々のチェーンと関連付けられる。層 k ランク・ファイル（ $1 \leq k \leq n$ ）の各レコードは、ランク値（0と $N_{k_{max}} - 1$ ）を受信するフィールド、およびランク・ファイルの次レコードのアドレスを受信するフィールドを有する。20
次アドレス・フィールドのデフォルト値（図示の例では0）は、レコードがチェーンの最後のレコードであることを意味する。ビットマップ・セグメント・ファイル（図15および図16の右側部分）は、関連するランク・ファイルと同じ方法でアドレス指定される。層 k のレコードごとに、これはランク・ファイルの対応するレコードに記憶されたランクと関連するビットマップ・セグメントを受信するため、 D_k ビットのビットマップ・フィールドを有する。ランク値および次レコード・アドレスは、同じファイルの2つのフィールドではなく、共通のアドレス指定を有する2つの別個のファイルに記憶してもよいことを理解されたい。

【0116】

V D Gコード化層 k ごとに、シソーラスの項目は、層 k ランク・ファイルのアドレスを含むための30
頭部アドレスを有し、ここに項目に関する第1のランク・レコードが記憶される。そこから、関連のランク・チェーンを検索することができる。例えば、 Max の層1ランク0、2および3（図13A）は、シソーラス項目（図14A）の頭部アドレス・フィールドで指示されたアドレス29で、次にチェーン・アドレス27および15で図15のランク・ファイルにアクセスすることによって検索される。並行して、対応する層1ビットマップ・セグメント001、010および100が読み出される。図15および図16は、ランクおよびビットマップ・セグメント・ファイルが、自由レコード（図15のアドレス32／33／17および図16の29／8／17／24）で構成された追加チェーンを有することも示す。後者のチェーンの頭部は、必要に応じて新しいコード化データを書き込むために割り当てられる。40

【0117】

シソーラス項目は、さらに、図14A、図14Gおよび図14Hの第3列で示すように、項目に属するチェーンの最終レコードのランク・ファイルにあるアドレスを含むため、層1末尾アドレス・フィールドを有することが好ましい。これによって、符号化データ記憶の更新が容易になる。例えば、対応する層1ビットマップ・セグメントで Max の新しい層1ランクを挿入する作業は、以下のように進行する。すなわち、自由レコード・チェーンの頭部を位置決めし（アドレス32）、次レコード・アドレス・フィールドにあるアドレス（33）が、新しい自由レコード・チェーン頭部のアドレスになり、アドレス32にあるレコードが、ランク・フィールドの新しい層1ランク、次アドレス・フィールドのチェーン終了フラグ（0）、およびセグメント・フィールドの新しいビットマップ・セグメ50

ントをそれぞれ受信し、Maxのシソーラス項目(15)の末尾アドレス・フィールドに含まれるアドレスが(チェーンの潜在的に長い経路を迂回して)直接アクセスされて、新しいデータのアドレス(32)を書き込み、これはMaxのシソーラス項目の末尾アドレス・フィールドにも書き込まれる。層1ランクがMaxにとって新しいランクであることは、層2のデータから決定することができる。すなわち、以前に実行した層2の更新が、層2ビットマップ・セグメントの「0」を「1」に変更し、層1ランクは、ワードによって新しく、それ以外の場合、層1ランクは既にMaxの層1ランク・リストに存在するので、前記層1ランクが見られるまで、これを取り消さねばならない。3つ以上の符号化層がある場合は、 $k > 1$ で、シソーラス項目に層k末尾アドレス・フィールドを提供し、層k+1データから決定された通り、新しい層kランクにて同じ方法で進行することが可能である。しかし、これを実行する際の主な利得は、チェーンが最も長い層1にある。

10

【0118】

図15および図16では、3つの異種シソーラス(顧客名シソーラス、事故金額ワード・シソーラスおよび事故金額マクロワード・シソーラス)から来るコード化データを同じデータ・コンテナに記憶する。図の明快さを期して、他のシソーラスは無視する。実際、1つの層のコード化データは全て、その由来となるワードまたはマクロワード・シソーラスに関係なく、同じランク/ビットマップ・セグメント・ファイルにパイルアップされる。これで、どのシソーラスのどの項目も、この2つの結合されたファイルにある個々のレコード・チェーンを指す。

【0119】

処理速度を最適化するため、レコードのグループを、それが関連するシソーラス項目に基づいてグループ分けするように、ディスク記憶についてランクおよびビットマップ・セグメントをソートすることが好ましい。これを実行する利点は、物理的に近接したレコードのグループの同時RAMローディングを可能にするコンピュータ・キャッシュ・メモリによって、1つのシソーラス項目のコード化データを読み出すのに必要なディスク・アクセス数が少なくなることである。その最適化ソートのバッチ実行は、シソーラス項目(頭部および末尾アドレス・フィールド)を同時に更新する必要がある、タイミングの悪いリソース使用を回避するために使用してもよい。

20

【0120】

この最適化を容易にするため、異なるシソーラスには、共通ファイルではなく別個のデータ・コンテナを使用することが好ましい。これによって、各回に記憶すべきデータの量が減少する。特に、シソーラスおよびコード化層ごとに1つのランク/ビットマップ・セグメント・ファイル対を使用するのが適当なようである。

30

【0121】

さらなる可能な方法は、異なるシソーラス項目に別個のランクおよびビットマップ・セグメント・ファイルを提供する。それには、メモリのファイル宣言数を多くしなければならない。しかし、処理速度の点から、上述した最適ソート作業を必要とせず、これが最適である。レコード・チェーンを指す頭部および末尾アドレスを記憶する必要もない。すなわち、シソーラス項目が、単に、ランクおよびビットマップ・セグメント・データが記憶されているデータ・コンテナを指定するのである。

40

【0122】

図17は、後者のケースで顧客名シソーラスのデータを構成する方法を示す。シソーラスは、シソーラス・ワードがソートされた状態にあるインデックス・レジスタを有する。ワードおよびコード化層kごとに、システム・メモリに2つのファイルが提供され、一方はランク・データを含み(NOkと呼ばれる)、一方はビットマップ・セグメントを含む(HPkと呼ばれる)。属性値(Andre、Arianeなど)を使用して、対応するファイルに命名することができる。記憶機構は、図15~図16に示すように、共通のデータ・コンテナほどコンパクトではないが、1つのワードに関するデータへのアクセスが、ソートの必要なく非常に迅速になる。

【0123】

50

図 17 に示すような構成は、オペレーティング・システムがメモリで管理できるファイル数に関する過度に過酷な制限に困っていない、または多数の個々のファイルの記憶によるオーバーヘッドが問題でない場合に好ましい。それ以外では、異なる（マクロ）ワード、または前述したように異なるシソーラスに関しても、ランクおよびビットマップ・ファイルをグループ分けすることが可能である。

【0124】

データ圧縮の高度化に加えて、多層行 ID リストの符号化方法は、大部分の問合せ処理を非常に加速させる。処理は、最初に、より高次の層で実施され、その結果が下層に渡される。コード化方式は、各層の全シソーラスの項目で共通の構造を維持し、仮想フラットファイルによって与えられた元の構造によって転写される。したがって、様々な層で、異なるシソーラスからの整数リストまたはビットマップ間の一括論理演算が可能である。層 $k+1$ で獲得された結果が、層 k での演算回数を最小にするためのフィルタのソートを提供し、これによって処理効率が特に複数アドレス問合せ基準にて強化される。

10

【0125】

この強化は、本明細書の簡略化した例では小さすぎて見えにくい。以下の要求を考察してみる。すなわち、金額が 1, 300 の Max の事故を検索する（応答がない）。直接の層 1 処理は、関連する層 1 のデータを読み出し、復号して、図 10A および図 10G のシソーラスにあるワード「Max」および「1, 300」のビットマップ・ベクトルを構築し、2つのビットマップ・ベクトルの論理 AND を計算する。層 2 で全く同じ種類の処理をすると、層 2 の方がレコードが少ないので、読み出し作業が少なくなり、ワード「Max」と「1, 300」に関する 2つの層 1 ランク・リスト（図 13A および図 13G の第 2 列）間に重複がないので、層 1 の処理が回避される。1, 300 ではなく 10, 000 の金額で同じ要求をされると、層 2 の結果は、ランク 0（他のランクがフィルタリングされてない）に対応する 2つの層 1 ビットマップ・セグメントをロードし、そのセグメント間の AND を計算するという層 1 の処理を減少させることができる。

20

【0126】

D1 および D2 のより代表的な値（例えば、 $D1 = D2 = 128$ ）および大きいサイズのデータベースでは、この 2 層間のフィルタリング原理は、優れた利得を提供する。層 2（層 $k+1$ ）でグループ全体のフィルタリングが達成されているので、層 1（または一般に層 $k \geq 1$ ）の処理から、大きいビットマップ・ベクトルが消える。

30

【0127】

VDG の作成および管理

図 18 は、本発明によるシステムのハードウェア・プラットフォームを形成するのに適したコンピュータ・システムの例示としてのレイアウトを示す。このハードウェアは従来のタイプでよい。以下のような複数のユニット間でデジタル信号を交換するため、バス 100 を有する。

- 中央演算処理装置（CPU）101；
- CPU の基本的演算命令を含む読出し専用メモリ（ROM）102；
- CPU 101 に作業スペースを提供し、CPU が扱うプログラム命令および変数を動的に含むランダム・アクセス・メモリ（RAM）103；
- 1つまたは複数の表示装置（または情報を人間に送出する他の種類の装置）を制御する回路、およびキーボード、マウス、デジタル・ペン、タッチ・スクリーン、オーディオ・インタフェース等のような取得装置からコンピュータ・システムへ情報を入力する回路を備えるマンマシン・インタフェース 104；
- RAM 103 にロードするデータおよびコンピュータ・プログラムを記憶する大容量記憶装置。図 18 に示す典型的な例では、大容量記憶装置は、1組の磁気ディスク 106 にデータを記憶するハード・ドライブ 105 を備える。本発明の実施には、磁気でも光学的でも、任意の種類の大容量記憶装置を使用することができることを理解されたい。

40

【0128】

本発明を実施するには、前述したようなデータ構造および以下でさらに詳細に説明するプ

50

プログラムを記憶するためにハード・ドライブ・ユニット105を使用する。プログラム命令および有用なデータが、処理するためにCPU101によって動的記憶機構RAM103にロードされる。問合せ結果は、ハード・ドライブに記憶したり、マンマシン・インタフェース104を通して、または遠隔アクセスの場合はネットワーク・インタフェース（図示せず）を通してユーザに配布したりする。

【0129】

大容量記憶装置105は、大量のデータの記憶に適しているが、アクセス時間はRAM103よりはるかに長い。これは、ハード・ドライブの読み取りヘッドを所望のディスク位置の前に運ぶために必要な時間のせいである。当技術分野で周知のように、ハード・ドライブ105でディスク・アクセスを実行する場合、実際に読み出されるデータは、ハード・ディスク上に近接して記憶されたデータのブロックを形成し、これは「キャッシュ」メモリと呼ばれるRAM103の部分にロードされる。CPUが同時に、または短い期間内に異なるデータ片を必要とする可能性が高いことが分かっている場合は、同じブロックに属するこのようなデータが、1回のディスク・アクセスで検索できるようにデータ記憶編成を構成すると都合がよく、これで処理時間が最短になる。

10

【0130】

図18のシステムとしては、デスクトップまたはラップトップ・タイプのパーソナル・コンピュータ（PC）を使用することができる。ワークステーションまたはメインフレーム・コンピュータを使用することもできる。

【0131】

もちろん、本発明の実施には、他のハードウェア・プラットフォームを使用してもよい。特に、当業者には、ビットマップ・セグメントおよびベクトルで実行される多くの計算は、専用論理回路またはコプロセッサによって効率的に実施されることを理解することができるだろう。さらに、このシステムでは並行計算が非常に自然である。

20

【0132】

次に、図19を参照しながら、最も一般的なデータ表現である図1～図3に示した形式である入力データ・テーブルからVDGデータ構造を作成するプロセスについて説明する。この作成プロセスは、レガシー・データベースからVDG構造を作成するのに適している。さらに説明されるVDG更新規則から、VDGを全く新しいデータから直接作成することができることを理解されたい。

30

【0133】

特定のデータベースでは、データ・テーブルは、図1～図3のような行IDではなく、複合キーを特徴とする自身の行を有する。複合キーは、データ・テーブルの幾つかのキー・フィールドの内容を連結したものである。ソース・データ・テーブルでは、レコードは、目標テーブルのレコードの複合キーを指定する外部キーを含む。このようなレガシー・データベースを扱う場合、VDG作成手順の第1段階は、複合キーを図1～図3に示す行IDのような単一キーに翻訳することである。この（任意選択の）第1段階を、図19のボックス120で示す。

【0134】

第2段階121は、適宜、空レコードでデータ・テーブルを完成することである。これは、図4～図7に関して上述したように、データ・テーブル木の各経路を経路の葉テーブルから根テーブルへと走査することによって実行される。空レコードへのリンクは、デフォルト値-1で表される。その結果、ソース／目標テーブル対ごとに、目標テーブルの全行IDが、ソース・テーブル・リンク列に少なくとも1回現れる。

40

【0135】

次の段階122は、ワード・シソーラスの作成を含む。関連する属性、すなわち、問合せ基準に使用される可能性が高い属性（全部でもよい）を決定する。決定した属性ごとに、ワード・フォーマット（タイプおよび長さ）を選択する。ワード・シソーラスごとに、関連する列で発生する属性値を、空値も含めて、ハード・ドライブ105に記憶されたデータ・テーブルから読み取る。反復する値は削除され、属性値および属性のタイプに適用する

50

ことができる順序関係に基づき、残りの値をソートする。このソート作業を、ハード・ドライブ105からCPUのキャッシュ・メモリに転送された連続データ・レコード・ブロックで実行し、各ブロックの処理後に、外部ソートする。

【0136】

次にVDG作成手順は、段階123の関連マクロワード・フォーマットの決定へと進む。マクロワード・シソーラスが生成されないワード・シソーラスもある（例えば、図10Cの顧客の性別シソーラスはマクロワードを必要としない）。これに対して、例えば、日付または金額属性に対応するシソーラスなど、異なる切り捨て長を有する幾つかのマクロワード・シソーラスを生成するシソーラスもある。属性列に見られる値が、アルファベットの大部分の文字で始まる文字ストリングを含む場合は、接頭語長が1文字のマクロワード・シソーラスを作成すると都合がよい。適切なマクロワード階層に関する決定は、データベース管理者によって実行され、マンマシン・インタフェース104を通して入力される。これは、シソーラスのワードの属性タイプおよび／または分布に基づく自動プロセスでもよい。段階123では、対応するワード・シソーラスのワードに切り捨てを適用し、反復するマクロワードを削除することにより、ソートされた形式で直接、マクロワード・シソーラスも作成される。

【0137】

マクロワード・シソーラスの各項目は、マクロワードの対象となる範囲に含まれる下位レベル・シソーラスの第1ワード（または下位レベルマクロワード）を示すことが好ましい。このように、考慮中のマクロワードと一致する接頭語を有する最下位ワード（またはマクロワード）を表示すると、そのマクロワードの「子」へのアクセスに必要な時間が短縮される。下位レベル・シソーラスを走査することなく、最初のものにアクセスできるからである。代替的または累加的に、接頭語がマクロワードと一致する最高位ワード（または下位レベルマクロワード）を、マクロワード・シソーラスで表示することができる。

【0138】

段階124では、リンク・テーブルの行および個々のワード・シソーラスの項目を作成する。これは、 $n = 2$ 層で符号化する場合、例えば、図20に示すアルゴリズムに従い、全フラットファイル（図8）を記憶せずに実行することが好ましい。

【0139】

図20～図32に示す実施形態では、属性値に関するシソーラスの各項目は、索引WIを含み、これは図14A、図14Gまたは図14Hに示すタイプの補助テーブルの行IDを形成し、図15および図16に示すタイプのコード化データ・コンテナを指す。符号化層kごとに、この補助テーブルは以下を有する。

- 層kに関連するコード化データ・コンテナに、索引WIのシソーラス・ワードに関する第1のレコードのAT_Fk (WI)と表示されたアドレスを含む列；
- 前述したように、データ・コンテナに索引WIのシソーラス・ワードのチェーンの最終レコードのAT_Lk (WI)と表示されたアドレスを含む列で、後者の列は、層1にしかないことがある。

【0140】

前述したように、任意のコード化層のデータ・コンテナを、シソーラスの全部または一部で共用するか、各個別シソーラスに関連付けることができる。層kコンテナのアドレスAD (≥ 1)にあるレコード（ここでは全シソーラスに共通とする）は、ランク・データを0から $Dk - 1$ の範囲の整数として含む第1のフィールドNOk (AD)、チェーンの次のレコードのアドレスNXk (AD)を含む第2のフィールド（このアドレスは、さらなるアドレスがない場合は0である）、および対応するビットマップ・セグメントHPk (AD)を含む第3のフィールドを備える。層kのコンテナは、第1のレコード・アドレスがHkと表記される自由レコード・チェーンを有する。

【0141】

補助テーブルも、別個のワード索引を含む幾つかのシソーラスで共用し、このような共通の補助テーブルにアクセスできることも理解される。

【0142】

段階124の前に、データ・コンテナの全レコードは、相互に連鎖されて自由であり、ビットマップ・セグメントHPk (AD) は、全てのゼロ・セグメントで初期化される。全ての補助テーブルの列AT__FkおよびAT__Lkも、値0で初期化される。

【0143】

フラットファイル行IDのD1でのユークリッド除法による商および剰余は、それぞれQ1およびR1と表示される。さらなる層 $k > 1$ ごとに、QkおよびRkはそれぞれ、Dkによる $Q(k-1)$ のユークリッド除法による商および剰余を指す。図20の初期化ステップ130で、整数Q1、R1、Q2およびR2は0に設定される。

【0144】

ハード・ドライブ105から1つずつ、またはブロックごとに読み出される根テーブルの行は、ステップ131で1つずつ選択される。選択した根テーブル行に連結された他のデータ・テーブルのレコードをステップ132で読み出す。これは、図5～図7にてコンパクトな形式で示したタイプのデータ・グラフを提供する。

【0145】

これらのデータ・グラフのリンク、すなわち、データ・テーブルの行IDを、行IDの $Q1 \times D1 + R1$ でリンク・テーブル(図9)の関連列に書き込む(ステップ133)。リンク・テーブルがない場合は、ステップ133を省略する。

【0146】

現データ・グラフについて、異なる属性ATを連続的に選択する(ステップ134)。選択した属性ATの値は、対応するシソーラスの二分検索によって位置決めされ、そのワード索引WIはステップ135で読み出される。ステップ136は、図21～図24に関して以下でさらに詳細に説明されるが、ワード索引WIのATシソーラス項目について、補助テーブルおよびデータ・コンテナを更新する。この更新は、現フラットファイル行IDの $Q1 \times D1 + R1$ をシソーラス・ワード索引WIに関連する整数リストに挿入することに相当する。

【0147】

全ての属性をこのように処理したら(テスト137)層1の剰余索引R1を、ステップ138で1ユニットだけ増加させる。増加したR1がD1に等しい場合(テスト139)は、索引R1を0にリセットし、層1の商索引Q1および層2の剰余索引R2をそれぞれ、ステップ140で1ユニットだけ増加させる。増加したR2がD2に等しい場合(テスト141)は、索引R2を0にリセットし、層2の商Q2をステップ142で1ユニットだけ増加させる。ステップ142の後、またはステップ139で $R1 < D1$ になるか、ステップ141で $R2 < D2$ である場合は、テスト143を実行して、全ての行または全ての根テーブルを考察したか判断する。考察していない場合、手順はステップ131に戻って、新しい根テーブル行を選択する。

【0148】

全ての根テーブル行を考察したら、図19の段階124が終了し、パラメータQ1、R1、Q2およびR2を、起こり得る新しいデータ・レコードをその後に挿入するため、記憶する。最終的に、仮想フラットファイルの行数は、 $Q1 \times D1 + R1$ で与えられる。

【0149】

図20の手順は、ステップ130で全てのQkおよびRkパラメータを0に初期化し、より高い層でステップ138からステップ142を発展させる(これはデータ・グラフ・ポインタ $Q1 \times D1 + R1$ を増分させることに等しい)ことにより、 $n > 2$ 符号化層に容易に拡張される。

【0150】

図21は、プログラムが属性ATに関するシソーラスのワード索引WIについて、データ・コンテナ内のレコード・チェーン、および層 $k \geq 1$ のシソーラス補助テーブルを管理する方法を示す。第1のステップ150は、補助テーブルに記憶されている値AT__Fk (WI) をアドレス変数ADにロードする。AD=0(テスト151)である場合は、シソ

10

20

30

40

50

ーラス索引 $W I$ についてレコード・チェーンを初期化しなければならず、したがってデータ・コンテナにある自由レコード・チェーンの頭部アドレス $H k$ が、ステップ 152 で $A T\_F k (W I)$ に指定される。

【0151】

シソーラス索引 $W I$ に既にレコード・チェーンがある場合（テスト 151 で $A D > 0$ ）は、ステップ 153 でランク $N O k (A D)$ をランク変数にロードする。その後のテスト 154 で、 q が商変数 $Q k$ と異なることが示されたら、ステップ 155 でアドレス変数 $A D'$ がチェーンの次レコードのアドレス、すなわち、 $N X k (A D)$ を受け取る。 $A D'$ がまだ 0 でない場合（テスト 156）、プロセスは、ステップ 157 で $A D$ を $A D'$ で置換した後、レコード・チェーンの次ランク変数を検査するため、ステップ 153 に戻る。テスト 156 で $A D = 0$ の場合、ステップ 158 で自由レコード・チェーンの頭部アドレス $H k$ が、次レコード・フィールド $N X k (A D)$ に書き込まれるように、データ・コンテナのレコードをシソーラス索引 $W I$ のチェーンに追加しなければならない。

【0152】

ステップ 152 またはステップ 158 の後、自由レコード・チェーンの頭部アドレス $H k$ をステップ 159 でアドレス変数 $A D$ にロードする。次にステップ 160 を実行して、補助テーブルおよびデータ・コンテナを更新する。この更新作業 160 は、以下で構成される。

- ー 頭部アドレス $H k$ を自由チェーンの次アドレス $N X k (A D)$ で置換し、
- ー アドレス変数 $A D$ の現値を $A T\_L k (W I)$ に書き込み、
- ー $Q k$ および 0 をそれぞれ、データ・コンテナのフィールド $N O k (A D)$ および $N X k (A D)$ に書き込む。

【0153】

ステップ 160 の後、または上述したテスト 154 で $q = Q k$ の場合は、ステップ 161 で、ビットマップ・セグメント $H P k (A D)$ を、そのセグメントのビット位置 $R k$ に数字「1」を書き込むことにより更新する。

【0154】

図 20 では、層 1 および層 2 のコード化データが両方ともステップ 136 で更新されると考えられた。すなわち、図 20 の手順は、 $k = 1$ で 1 回、 $k = 2$ で 1 回実行される。 $k = 1$ でしか実行せず、その後に段階 124 で生成された層 1 のランク・データを処理することにより層 2 のコード化データを作成するという可能性もある。

【0155】

図 20 のように、レガシー・データベースから $V D G$ を初期化する場合、ランク・データ $Q k$ は小さい順に表示されることに留意されたい（図 21 のテスト 154 では、常に $q \leq Q k$ である）。したがって、ステップ 150 で、 $A D = A T\_F k (W I)$ ではなく、レコード・チェーンの末尾、すなわち、 $A D = A T\_L k (W I)$ に直接移動させることが可能である。この場合、テスト 154 で $Q k > q$ の場合は、ステップ 158 を直接実行し、レコード・チェーンの走査を回避する。このような実施形態を図 22 に示す。

【0156】

後者の実施形態では、 $V D G$ の初期化が終了したら、 $k > 1$ の状態で、層 k の末尾アドレス・フィールド $A T\_L k$ を廃棄してもよい。しかし、挿入される可能性が高い新しい $V D G$ が、既存の $V D G$ のフラットファイル行 $I D$ 全部と等しい、またはこれより大きいフラットファイル行 $I D$ を有するよう（すなわち、削除した $V D G$ のフラットファイル行はもう使用しないよう）、 $V D G$ を管理している場合、その後の更新を全て、図 22 の実施形態に従って実行するため、全ての末尾アドレス・フィールド $A T\_L k$ を保持することが有利である。

【0157】

図 21 に示す形式では、更新手順が別個にランク値 $Q k$ の仮説に適用することができる。

【0158】

図 23 および図 24 は、ステップ 136 でワード索引 $W I$ について $A T$ シソーラス項目に

関する補助テーブルおよびデータ・コンテナを更新する代替方法を示し、これは層1 ($n = 2$ のコード化層を有する) の補助テーブルの末尾アドレス・フィールド AT_L1 を利用する。図23は、図24の層1の処理より前に実行される層2の処理を示す。図23～図24のステップは大部分が、図21のステップと非常に似ており、したがって対応する参照番号を使用している。

【0159】

図23の層2の処理は、基本的に図21 ($k = 2$) と同じであるが、以下の違いがある。

— ステップ160で、末尾アドレス・フィールド AT_L2 (WI) を処理する必要がない。

— ステップ161が、さらに、2進変数 $LL1$ を「1」に設定することを含み、これは 10
、現層1のランク・データ $Q1$ が、ワード索引 WI に関する層1のレコード・チェーンに属していないことを意味する。

— テスト154で $q = Q2$ の場合は、別のテスト164を実行して、層2セグメント $HP2$ (AD) のビット位置 $R2$ が値「1」を含むか判断し、テスト164がノーの場合のみ、その後にステップ161を実行する。

— テスト164がイエスの場合は、現層1のランク・データ $Q1$ が既に、ワード索引 WI に関する層1のレコード・チェーンに属し、したがってステップ165で変数 $LL1$ を「0」に設定する。

【0160】

図24の層1の処理は、ステップ170で、 $LL1$ が0か1かをテストして開始する。 $LL1 = 0$ の場合は、ステップ150を実行して、層1の補助テーブルに記憶されている値 AT_F1 (WI) をアドレス変数 AD にロードし、ループ153～155を実行して、ランク $Q1$ に関するデータが記憶されているデータ・コンテナアドレス AD を求める。ステップ153およびステップ154は、図21と同じであり、ステップ155で、次アドレス $NX1$ (AD) をアドレス変数 AD に直接ロードする ($LL1 = 0$ であるので、 AD が0になることはない)。テスト154で $q = Q1$ の場合、プログラムはステップ161へと進む。 20

【0161】

テスト170で $LL1 = 1$ の場合は、ステップ171を実行し、層1の補助テーブルに記憶されている値 AT_L1 (WI) をアドレス変数 AD にロードする。 $AD = 0$ の場合 (テスト172) は、図21と同様に一連のステップ152、159から161を実行する (しかし、ステップ160で次アドレス・フィールド $NX1$ (AD) を扱う必要はない)。テスト172で $AD = 1$ の場合は、図21と同様に一連のステップ158～161を実行する。 30

【0162】

図23～図24の手順は、ランク・データ $Q1$ が層1のレコード・チェーンにない場合、ランク値の仮説なしで、このようなチェーンを走査しない。

【0163】

個別ワード・シソーラスのコード化データを全て作成した後、図19に示す手順の次の段階125は、記憶したコード化データを再構成する。前述したように、これを実行して、 40
各層のコード化データ・コンテナのレコード・チェーンを編成し、したがって同じシソーラス・ワードに属するレコードは、CPUのキャッシュ・メモリによって1回または数少ないディスク・アクセスでアクセス可能になるよう、連続アドレスを有する。これを実行するための簡単な方法は、新しい補助テーブルおよび新しいコード化データ・コンテナのためにメモリ・スペースをとっておくことである。シソーラス・ワードを1つずつ考察し、それぞれで古い補助テーブルで指示されたコード化データを順次読み出し、書き込み作業ごとに増分されたアドレス AD で、新しいデータ・コンテナにコピーする。次のシソーラス・ワード索引 $WI + 1$ へと進んだら、新しいポインタ AT_Lk (WI) = $AD - 1$ および AT_Fk ($WI + 1$) = AD を決定し、新しい補助テーブルに記憶する。全てのコード化データ・レコードをこのように読み出して、新しいデータ・コンテナに再度書き 50

込んだら、古いデータ・コンテナおよび補助テーブルを廃棄する。

【0164】

このような再構成は、コード化層 k ごとに別個に実行することができる。

【0165】

コード化層のシソーラスによって幾つかのデータ・コンテナがある場合は、これを別個に発注してもよい。

【0166】

前述したように、シソーラスが、図17で示した方法で編成されている場合は、再構成ステップ125が省略される。コード化データ・ファイルが、当然、シソーラス・ワードに関するグループ化状態を満足しているからである。

【0167】

図19に示した手順のその後の段階126では、マクロワード・シソーラス項目を作成する。マクロワードおよび層ごとに、これはマクロワードがカバーするワード（または下位レベルマクロワード）のランク・コード化データ Q_1 、 Q_2 をマージし、ワード（または下位レベルマクロワード）に関するものの論理ORによって対応するビットマップ・セグメントを獲得することにより、簡単に実行される。段階125で示すように、ワード・シソーラスに関してコード化データが再構成されている場合は、マクロワード・シソーラスでコード化データの同じグループ化が自動的に達成される。

【0168】

段階127では、元のデータ・テーブル（図1～図3）でもう無用になったリンク列を削除することができる。段階121で追加された空レコードも削除することができ、その発生は、リンク・テーブル（図9）のデフォルト値-1で表示される。

【0169】

最後に、上述した実施形態でハード・ドライブ105に記憶される要素は以下の通りである。

ー リンク列がない、図1～図3で図示したようなデータ・テーブル。図4のデータ・テーブル木構造を定義するパラメータは、テーブルとともに記憶される。

ー 図9で図示したようなリンク・テーブル。

ー 所望の各属性について、指標レジスタおよび補助テーブルを備え、記憶されているシソーラス。図25～図26は、本明細書の簡略化した例の属性 $AT = CN$ （「顧客名」）および $AT = AA$ （「事故金額」）の指標レジスタを示す。図28～図29は、対応する補助テーブルを示す。

ー 特定の切り捨てまたは接頭語長を有し、個別ワード・シソーラスのように編成されたマクロワード・シソーラス。各マクロワード・シソーラスの指標レジスタは、さらに、マクロワードごとに、同じ属性の下位レベルのシソーラスの指標レジスタにマクロワードの対象となる第1ワード（またはマクロワード）の行IDを含む追加列を有する。図27および図30は、属性 $AT = CN$ および切り捨て長3の指標レジスタおよび補助テーブルを示す。

ー それぞれが自身の自由レコード・チェーンの変数頭部アドレスを有する各コード化層のコード化データ・コンテナ。図31および図32は、図24～図29のシソーラスが共用する層1および層2のデータ・コンテナ（それぞれ自由レコード・チェーン頭部アドレス31および27）を示す。

ー 任意選択で、高密度の属性に適した「低密度」フォーマットで記憶された1つまたは複数のシソーラス。低密度フォーマットでは、 $n = 0$ およびフラットファイル行IDを、明示的（短）整数リストとして、例えば、レコード・チェーンによって記憶する。層1から n のコード化データが必要な場合、これは、リストに記憶された各整数から連続的にユークリッド除法を n 回実行することにより、容易に計算される。所与の高密度属性では、個々のワード・シソーラスを低密度フォーマットで、1つまたは複数のマクロワード・シソーラスを「通常の」符号化したフォーマットで提供することが適当である。

【0170】

10

20

30

40

50

図 3 1 および図 3 2 のデータ・コンテナは、再構成段階 1 2 5 に従って図 1 5 および図 1 6 に示すコンテナから取得し、ここで図 2 8 ~ 図 3 0 の補助テーブルも、それぞれ図 1 4 A、図 1 4 G および図 1 4 H から獲得される。A n d r e と呼ばれる顧客に関する問合せを処理するには、プロセッサは、コード化データ・コンテナが再構成されていなければ、図 1 5 のレコード # 2 0、# 1 1 および # 2 を読み出さなければならず（自身を層 1 に制限する）、一方で図 3 1 の再構成されたコンテナの物理的に連続するレコード # 1、# 2 および # 3 を読み出す。後者の読み出しは、通常、1 つのブロックをキャッシュ・メモリにロードすることによって実行することができる。分散したレコードを読み出すには、ディスクへのアクセスを増加させる必要があり、したがって処理時間が長くなる。

【0 1 7 1】

10

図 9 および図 2 8 ~ 図 3 2 の破線で分離された列は、別個に記憶することが好ましい。例えば、このような列にある 1 つの値の記憶アドレスは、列に割り当てられた開始アドレスに、その行 I D に等しいオフセットを加え、フィールド長を掛けた値と定義することができる。

【0 1 7 2】

したがって、リンク・テーブル行のリンク（データ・グラフ）フラットファイル行 I D によって与えられた対応のアドレスに記憶される。このリンク列の別個記憶は、問合せで要求された出力データを生成するためにデータ・テーブルの一部にアクセスする必要がない場合、データ検索を加速する。

【0 1 7 3】

20

同様に、問合せ処理で実行される基本的作業には、1 層のコード化データしか必要としないものがあり、したがって処理を加速するために補助テーブルの各層に関する情報を分離すると都合がよい。ランク・データとビットマップ・セグメント・データを別個に示唆する作業もあり、したがって図 3 1 および図 3 2 で示すようにデータ・コンテナ内でこれらのデータを分離すると有利である。

【0 1 7 4】

シソーラスを記憶する代替方法では、ワード指標レジスタおよび補助テーブルを、ハフマン・タイプの索引付けで 1 つのテーブルにマージする。すなわち、そのテーブルの各行が属性 A T の値、A T _ F k および A T _ L k データ、次行ポインタ（次行は記憶されたシソーラスにある属性の次の値を含む）、および任意選択で先行行ポインタを含む。

30

【0 1 7 5】

実施形態では、上述したように作成された V D G の保守は、以下の作業を含む。

【0 1 7 6】

1 / レコードの挿入

新しい仮想データ・グラフ、すなわち、フラットファイルの新しい行は、概ねデータ・テーブルへの新レコードの挿入に応答して作成される。

【0 1 7 7】

しかし、新レコードが、別の目標テーブルの既存レコードとのリンク有し、したがって、前記既存レコードを指す他のリンクがない場合は、新しいデータ・グラフは必要ないが、既存データ・グラフを更新する必要がある。例えば、顧客 O s c a r が、例えば、自動車について、新しい保険に加入したら、新しい V D G を作成せずに、保険データ・テーブルに新しいレコードを追加する。すなわち、図 7 のデータ・グラフを単に修正して、新しいデータを保険テーブルに対応するノードに入れる。次に、O s c a r が、例えば、住宅について第 2 の保険に加入したら、新しい V D G が必要になる。

40

【0 1 7 8】

新しい V D G を作成するには、空レコードを含め、挿入された新しいレコードに関して他のデータ・テーブルからの全レコードを、必要に応じて個々の行 I D によって識別し、これはこの関連レコードの属性値に基づき、問合せで検索することができる。

【0 1 7 9】

新レコードをデータ・テーブルに追加した後、最初にすることは、新しい属性値が発生し

50

た場合、必要になりそうな全ての新シソーラス項目を初期化することである（全ての A T _ F k および A T _ L k フィールドは 0 に初期化される）。新しい仮想フラットファイル行およびその対応するシソーラス項目は、図 20 のステップ 1 3 3 から 1 4 2 のように作成することができる。より高いレベルのマクロワード・シソーラスを、これに応じて更新する。

【0180】

2 / レコード属性の修正

既に存在するデータ・テーブルの属性値と変更または追加しても、リンク・テーブルに影響を与えず、これはテーブル内容を反映しないが、リンク構造は反映する。追加は、先行属性値が空だった場合の変更の特殊ケースである。同様に、レコードからの属性値の削除は、新しい属性値が空であった場合の変更の特殊ケースである。

10

【0181】

新しい属性値が新しいシソーラス項目を必要とする場合、このような項目を初期化する（A T _ F k = A T _ L k = 0）。修正すべきデータ・レコードを備えるフラットファイル・レコードに対応するリンク・テーブル行 I D のリスト L は、適切な問合せをすることによって獲得される。後者のリスト L を、新しい属性値のフラットファイル行 I D のリスト L ' とマージし（ビットマップ O R 演算）、マージしたリスト L ∨ L ' のコード化データを新しい属性値に割り当てる。リスト L の補数

【0182】

【数7】

20

$\bar{L}$

も、先行属性値の単層行 I D リスト L " と交差する（A N D e d）ように決定される（ビットマップ N O T 演算）。その結果の交差リスト

【0183】

【数8】

$\bar{L} \wedge L''$

が空白でない場合は、そのコード化データを先行属性値に割り当てる。これは、以前に先行属性値に関連するレコード・チェーンに属していた 1 つまたは複数のデータ・コンテナの自由レコード・チェーンへの転送でよい。交差リスト

30

【0184】

【数9】

$\bar{L} \wedge L''$

が空白である場合、先行属性値をそのワード・シソーラスから削除することができる。より高いレベルのマクロワード・シソーラスがあれば、同じ交差および更新シーケンスを実行する。

40

【0185】

3 / レコード・リンクの修正

ソース・データ・テーブルのリンクを変更すると、リンク・テーブルにあるリンクの出現に対応する変化が生じる。関係するリンク・テーブル行のリスト L は、適切な問合せを処理することにより決定することができる。

【0186】

前のリンクによって指示された目標テーブル・レコードが、これを指示するリンクをこれ以上有さない（その行 I D が、修正後にリンク・テーブルの対応する列にもう発生しない）場合は、新しい V D G が作成される。修正したリンクの下流では、この新しい V D G は、修正中のものと同じ内容を有する。修正したリンクの上流では、これは空レコードで構

50

成される。新しい仮想フラットファイル行およびその対応するシソーラス項目は、図 20 のステップ 133 から 142 のように作成することができる。より高いレベルのマクロワード・シソーラスがあれば、これに応じて更新される。

【0187】

その後、前のセクションで説明したものと同様の手順を、目標テーブルの各属性について実行することができる。すなわち、／a／リスト L を、新しい属性値（新しいリンクによって指示された目標テーブル・レコードで発生した値）のフラットファイル行 ID リスト L' とマージし、／b／マージされたリスト L ∨ L' のコード化データを新しい属性値に指定し、／c／リスト L の補数

【0188】

【数 10】

┌

を先行属性値（以前のリンクによって指示された目標テーブル・レコードで発生した値）のフラットファイル行 ID リスト L'' と交差させ、／d／結果の交差リスト

【0189】

【数 11】

┌へ┐

のコード化データを先行属性値に指定し、／e／より高いレベルのマクロワード・シソーラスがあれば、その全てで同じ交差および更新シーケンスを実行する。

【0190】

（修正したリンクの）第 1 の目標テーブルが、第 2 の目標テーブルへのリンク列を有する場合、第 2 の目標テーブルに関連したリンク・テーブルの列、およびリスト L の各行に記憶されたリンク値も変更し、第 2 の目標テーブルの属性ごとに、上述した手順／a／から／e／を実行する。これを、データ・テーブル木の第 1 の目標テーブルの下流に位置する全てのデータ・テーブルで反復する（図 4）。

【0191】

例えば、図 3 の事故テーブルで補正を実行し、事故 # 6 が保険 # 0 ではなく保険 # 2 で扱われる、すなわち、A r i a n e の自動車保険ではなく M a x の住宅保険に関することを示す場合、図 5 のデータ・グラフにある事故テーブルからのリンクを、保険レコード # 2 を指示するよう変更し、保険テーブルからのリンクを顧客レコード # 4 を指示するよう変更しなければならない。新しい行を仮想フラットファイルに挿入し事故が発生していない A r i a n e の自動車保険に関する有用な情報を含める。その結果、図 9 のリンク・テーブルの行 # 6 は、顧客、保険および事故リンク列にそれぞれ 4、2 および 6 の値を含むよう変更され、それぞれ 2、4 および -1 の値を含む新しい行 # 12 が追加され、これに応じてシソーラスが変更される。

【0192】

4／ レコードの取り消し

根テーブルからのレコードの取り消しは、そのレコードを含むフラットファイルの行を削除する。対応するフラットファイル行 ID が、シソーラスで符号化されたリストから除去される。すなわち、ビットマップ・ベクトルの関連付けられた位置にゼロが書き込まれる。このフラットファイル行 ID を、例えば、上記のセクション 1／または 3／に従って、さらなる V D G 挿入をできるようにする。仮想フラットファイルのサイズが主要な関心事でない場合、これは空白行のままにしてもよい。同様に、対応するソース・テーブルでこれを指示するリンクがない目標テーブルからレコードを取り消すには、そのレコードを含むフラットファイルの行を削除する（これらの行は、取り消されるレコードの上流にて、空レコードでデータ・グラフを表していた）。

【0193】

10

20

30

40

50

取り消すレコードが、強制的リンク（例えば、本明細書の例の顧客または保険テーブル）の目標テーブルに属している場合、そのレコードを含むフラットファイル行も削除される。取り消すレコードが、任意選択リンク（例えば、図4の例の第三者または仲介者テーブル）の目標テーブルに属している場合、取り消しはリンクの修正を含み、これによってそのレコードを指示するリンクが全て、空レコード（リンク値＝－1）へのリンクで置換される。このような修正は、上記のセクション3／で説明したように実行することができる（しかし新しいVDGは生成しない）。

【0194】

リンク・テーブルの対応する列にそれ以上行IDが発生しない非空目標テーブル・レコードを指すリンクが、取り消したレコードにある場合、その全てで、前記非空目標テーブル内およびその下流に、取り消したレコードと同じデータを、取り消したレコード内およびその上流に空値を含む新しいVDGを作成する必要がある。新しい仮想フラットファイル行およびその対応するシソーラス項目を、図20のステップ133から142のように作成することができる。より高いレベルのマクロワード・シソーラスがあれば、これに応じて更新される。

【0195】

5／ シソーラスの更新およびソート

シソーラス項目の上述の構造では、そのフラットファイル行IDリストが空白になった場合に発生するシソーラスのワードの取り消しは、シソーラス項目のHPkデータをゼロに維持することによって実行できる。しかし、メモリの使用に関しては最適でない。

【0196】

より効率的な方法は、取り消したワードの項目WIについて補助テーブルがAT__Fk（WI）＝AT__Lk（WI）＝0を有するように、データ・コンテナのレコード・チェーンを更新することである。このようなケースでは、ワード索引WIを解放することができ、デフォルト値（例えば、－1）が、シソーラス指標レジスタの取り消したワードについて、ワード指標コラムに書き込まれる。

【0197】

新しいワード・シソーラス項目の作成は、図21～図24に示すように実行することができる（テスト151または172でAD＝0）。ワード索引WIは、シソーラス項目の数を表すカウンタを増加させるか、使用できるワード索引（例えば、別のワードを取り消した時に、以前に解放されたワード索引）を選択することによって獲得される。このプロセスでは、行ID＝WIで、対応する属性の補助テーブルに、（有用な）行を追加する。

【0198】

マクロワード・シソーラスの更新に、同様の手順を適用することができる。マクロワードの取り消し時に、マクロワード索引WIを解放することができる（その構成ワードは全て取り消されている）。ワード作成の場合は、マクロワードが既に存在するか最初にチェックし、存在する場合は、そのマクロワード索引WIが回復され、それ以外ではマクロワードも作成される。

【0199】

したがって、ワードを除去および／または追加したら、補助テーブルはもうシソーラス・ワードの昇順でソートされない。シソーラスのソートを維持するため、ワード指標レジスタを操作しなければならない。

【0200】

しかし、このようなワード指標レジスタの操作を迅速に実行する必要がない。多少時間がかかるシソーラス全体のソート作業を必要とすることなく、更新したデータベースを新しい問合せに使用できるので、これは非常に有利である。

【0201】

シソーラスの新たに作成したワードまたはマクロワードは、別個の2次指標レジスタに記憶された自身のワード索引を有することができるが、これはシソーラスの前のワードと同じ補助テーブルおよびコード化データ・コンテナを共用する。シソーラスの追加時には、

この2次指標レジスタのみをソートすることができ、これはシソーラス・ワードの大部分が1次レジスタに属しているので、比較的軽い仕事である。ワードを削除する場合、1次または2次指標レジスタにあるその行は、ワード索引列にデフォルト値が残る。したがって、任意のワード範囲に関するコード化データにアクセスするには、1次および2次指標レジスタの両方で二分法によって範囲の境界を探して、関連するワード索引を決定し、次にこれを通常の方法で使用して、共通の補助テーブルおよびデータ・コンテナをアドレス指定する。

【0202】

時々、CPU101が使用可能な場合に、バッチ作業を実行して1次および2次指標レジスタをマージしながら、ワード指標列にデフォルト値を有するその行を削除することがある。これは、両方のレジスタが既にソートされているので、簡単な外部ソート作業である。それで生成されたマージ済みレジスタをセーブして、1次レジスタと置換し、2次レジスタを取り消す。

【0203】

このようなマージ作業を実行する前に、2次ワード指標レジスタが大きくなりすぎた（すなわち、項目を追加するたびに、ソートに時間がかかりすぎる）場合は、さらなる3次指標レジスタを作成し、新しいシソーラス項目を受け取り、以下同様にすることが可能である。

【0204】

6／ データ・コンテナの最適化
シソーラス編成が、図17に示したタイプではなく図25～図32に示したタイプである場合に、これは有用である。

【0205】

レコードがコード化データ・コンテナ内で挿入、削除されるので、同じシソーラス・ワードに属するレコードが連続アドレスを有するように、レコード・チェーンを配置することが好ましいという上述の条件はもう満たされない。これは、データベース・システムの満足できる動作を妨げるものではない。しかし、問合せ処理時間を最適化するため、上述したステップ125のように、コード化データ・コンテナのレコード、およびシソーラス補助テーブルの対応する列を再構成することが好ましい。ワード指標レジスタのソートと同様、このような再構成はCPUの時間が使用可能である場合に実行することができる。

【0206】

代替シソーラス構成

シソーラスを、図17で示した好ましい編成に従い、ワードまたはマクロワードごとに別個のファイルにして構成する場合、図19～図24の流れ図が多少簡単になる。第一に、図19の段階125を実行しない（VDGを作成し、保守する場合、ファイル編成は、このようなソートを省略すると有利である）。図20では、ステップ136の二分検索135およびシソーラス更新を、図33に示した手順と置換することができる。

【0207】

この手順では、 $\text{imax}(AT, W, k)$ が、シソーラスATおよびワードWに関するコード化データ・ファイルの層kレコードの現数を指定する。これらのパラメータは、初期化ステップ130でAT、Wおよびkの全ての値でゼロに設定される。

【0208】

図20のステップ134で選択した属性ATの現データ・グラフの値を図33のステップ175で変数Wに割り当て、コード化層索引kを1に初期化する。コード化データ・ファイルのレコードを指示する整数iは、最初にステップ176でゼロに設定される。その後のテスト177で $i = \text{imax}(AT, W, k)$ である場合、値 Q_k を有するレコード $AT\_W\_NO_k(i)$ を、ワードWに属する層kランク・ファイルに追加し、全ゼロ値を有するレコード $AT\_W\_HP_k(i)$ を、対応するビットマップ・セグメント・ファイルに追加する。これはステップ178で実行され、ここでは $\text{imax}(AT, W, k)$ も1ユニットだけ増加する。テスト177で $i < \text{imax}(AT, W, k)$ である場合は、

ステップ179でランクAT_W_NO_k(*i*)をランク変数 q にロードする。その後のテスト180で q が商変数 Q_k と異なることが判明した場合は、ステップ181で整数 i を1ユニットだけ増加させ、ファイルに次のランク変数がある場合は、それを検査するため、プロセスはステップ177に戻る。したがって、各層 k のコード化データ・レコード・チェーンの走査(図21のループ153~156に対応する)が、AT_W_NO_kファイル内で実行され、これはシソーラスの全ワードに共通のデータ・コンテナより小さい。したがって、最小数のディスク・アクセスが確保される。

【0209】

ステップ178の後、またはテスト180で $q = Q_k$ である場合は、関連するコード化データ・ファイルにあるビットマップ・セグメントAT_W_HP_k(*i*)のランク R_k のビットに「1」を書き込む(ステップ182)。テスト183で、コード化層索引 k を n (その後により高い層のコード化データを計算する場合は、これより低い値)と比較する。 $k < n$ なら、ステップ176に戻る前に、ステップ184で索引 k を1ユニット増加させる。 $k = n$ なら、シソーラスの更新が終了し、プログラムは図20のステップ137へと進む。

【0210】

図33の手順では、それぞれが整数値で構成されるランク・データAT_W_NO_kを大きいブロックでハード・ドライブ105からキャッシュ・メモリに読み出すことができ、したがってこの手順は非常に高速である。

【0211】

シソーラスで利用できる別の選択肢は、ワードに関する各項目に、フラットファイル行IDリストの代表的フォーマットの表示を含めることである。実際、フォーマット(例えば、低密度または通常密度)を、考察中のワードを含むデータ・グラフの数に応じて、ワードごとに選択することができる。これは、フォーマットが2つしかない、すなわち、低密度(0)および $n = 2$ コード化層(1)の通常密度である場合、図25~図27の右側部分にて破線で図示されている。この例では、シソーラス項目が全て、通常密度のフォーマットである。フォーマットが3個以上あってもよく、例えば、シソーラスのフォーマット・データは、ワードごとにコード化層の数を指定することができる。フラットファイル行IDリストが、1つまたは複数のシソーラスに共通なデータ・コンテナに記憶されたデータで表される場合、異なるコード化フォーマットに別個のコンテナを設ける。

【0212】

上述した選択肢を使用する場合は、記憶機構を最適化するため、データベースが生きるので、各シソーラス項目のフォーマットを修正することができる。問合せエンジンはどのフォーマットでも働くことができるので、これは優先順位が低いタスクである。例えば、シソーラス項目を更新している場合、修正したデータベースのワードの密度によって満たされた所定の条件に基づき、フォーマット変更が望ましいと思われる項目をマークすることが可能である。例えば、そのシソーラス項目で特定数のデータ・グラフが識別された場合、ワードまたはマクロワードを低密度から通常密度のフォーマット、およびその逆に変更することができる。その後、プロセッサの時間が使用可能な場合、マークした項目を新しいフォーマットに変換し、システムを最適化することができる。

【0213】

以上で、 $n > 1$ である場合は、フラットファイル行IDリストが、全層のビットマップ・セグメント・データおよび最終層のランク・データによって完全に定義されているので、各コード化層でのランク・データの記憶が多少冗長であると述べた。

【0214】

図34A~図34Bは、コード化データ・ファイルを構成する代替方法を示し、これは $k < n$ の層 k ランクを記憶しない。この構成では、補助テーブル(図28~図30)が層 n のデータ・コンテナの第1のレコードを指示することが重要である。すなわち、アドレスAT_F1およびAT_L1が必要ない。図34Aに示す最高層 $n = 2$ のデータ・コンテナは、図32のものと同一であり、下位層 $n - 1 = 1$ のデータ・コンテナにあるレコード

・チェーンの頭部アドレス $F(n-1)(AD) = F_1(AD)$ を含むため、各レコードに追加フィールドがある。後者のデータ・コンテナ(図34B)は、データ・コンテナによってカバーされる各シソーラス項目に関する層 n ランクごとに1つのレコード・チェーンを有する。層 $k < n$ データ・コンテナの各レコードは、チェーンの次レコードのアドレス $NX_k(AD)$ を含む第1のフィールド(このアドレスは、さらなるアドレスがない場合は0である)、および対応するビットマップ・セグメント $HP_k(AD)$ を含む第2のフィールドを有する。層 $k < n$ チェーンは、チェーンの頭部アドレスを含む上位層データ・コンテナのレコードに記憶されるビットマップ・セグメント $HP(k+1)$ の非ゼロ・ビットに従って順序づけられる。 $1 < k < n$ (図示せず) の場合、レコードは、さらに、下位層 $k-1$ のデータ・コンテナにあるレコード・チェーンの頭部アドレスを含む第3のフィールドを有する ($k=1$ になるまで以下同様である)。

【0215】

図34A～図34Bのデータ・コンテナを指示するシソーラスからフラットファイル行IDリストを検索する手順は、以下の通りである。ワード索引WIを使用して、層2データ・コンテナにある第1の関連レコードのアドレスを取得する。このアドレス(および NX_2 フィールドによって定義されたチェーンの各アドレス)で、層2ランク NO_2 を読み出し、ビットマップ・セグメント HP_2 を走査する。この走査で「1」が発見されるたびに、ビット位置 R_2 で、層1のランク $NO_1 = NO_2 \times D_2 + R_2$ を決定し、下位層データ・コンテナの対応するレコードを読み出す(最初は、層2データ・コンテナの列 F_1 によって与えられた頭部アドレスで、次に層1データ・コンテナの NX_1 アドレスによって指示されたアドレスで)。この方法により層1のビットマップ・セグメント HP_1 およびその位置 NO_1 を検索して、所望のフラットファイル行IDリストを表すビットマップ・ベクトルをアセンブルする。

【0216】

一般的ケースでは、データ・コンテナを層 n からアクセスする。 $k > 1$ のランク NO_k を決定した後に読み出した各セグメント HP_k を走査して、その非ゼロ・ビットを検出する。位置 R_k で検出された HP_k の各非ゼロ・ビットは、下位層ランク $NO(k-1) = NO_k \times D_k + R_k$ を提供し、下位層コンテナで指定されたチェーンで対応するビットマップ・セグメント $HP(k-1)$ が読み出される。 $k=1$ になるまでプロセスを再帰的に反復する。数字 $NO_1 \times D_1 + R_1$ は、シソーラス項目のフラットファイル行IDである。

【0217】

図34A～図34Bに示したコード化データ・ファイルは、図19～図21に関して説明したものと同様の方法で作成することができる。全ての HP_k および $F(k-1)$ フィールドは、段階124の前にゼロで初期化される。図21の手順は、 $k=n$ でのみ実行され、ステップ161ではコード化層索引 k が n から1に減少する図35のループによって置換される。

【0218】

このループの第1のステップ450は、ビットマップ・セグメント $HP_k(AD)$ のビット位置 R_k に数字「1」を書き込む。コード化層 k が1より大きい場合(テスト451)、これはステップ452で1ユニットだけ減少させ、層 $(k+1)$ コード化データ・コンテナで第1のアドレス $M = F_k(AD)$ を読み出す(ステップ453)。

【0219】

M がゼロである場合(テスト454)、層 k コード化データ・コンテナの自由レコード・チェーンの頭部アドレス H_k を層 $(k+1)$ コード化データ・コンテナの第1のアドレス・フィールド $F_k(AD)$ に書き込み(ステップ455)、新しいチェーンを作成する。次に、 AD の値を H_k で置換し(ステップ456)、層 k コード化データ・コンテナでレコード・チェーンを更新する(ステップ457～458)。 $NX_k(AD)$ を0に設定する前に H_k を $NX_k(AD)$ で置換する。ステップ458の後、プロセスはステップ450に戻る。

【0220】

テスト454で $M > 0$ の場合は、ステップ460で索引 R を0に設定して、ビットマップ・セグメント $HP = HP(k+1)(AD)$ の走査を初期化する。 R が、現データ・グラフ識別子に対応する剰余 $R(k+1)$ より小さい場合は、ビットマップ・セグメント HP の対応するビット $HP(R)$ を評価する(テスト462)。 $HP(R) = 0$ の場合は、プログラムがステップ463へと進み、テスト461に戻る前に R を1ユニットだけ増加させる。テスト462で $HP(R) = 1$ である場合は、層 k レコード・チェーンを前進する必要がある。整数 M' が、ステップ464で M の値を受け取り、ステップ465で M が $N \times k(M')$ に置換される。 M の新しい値がゼロでない場合(テスト466)、プログラムは上述したステップ463へと進む。それ以外は、層 k レコード・チェーンの最後に到達しているので、ステップ467で層 k の自由レコード・チェーンの頭部アドレス H_k を $N \times k(M')$ に割り当ててから、上述したステップ456へと進む。 10

【0221】

テスト461で R が剰余 $R(k+1)$ と等しい場合は、ビットマップ・セグメント HP の対応するビット $HP(R)$ も評価する(テスト470)。 $HP(R) = 1$ の場合は、現シソーラス項目に関係する層 $k+1$ の入力リストに既にランク Q_k が存在し、したがって層 k コード化データ・コンテナに新しいレコードを作成する必要がない。 AD の値は、ステップ471で単に M で置換され、プロセスはステップ450へと戻る。

【0222】

テスト470で $HP(R) = 0$ の場合は、 AD の値を自由レコード・チェーンの頭部アドレス H_k で置換し(ステップ472)、層 k のコード化データ・コンテナでハフマン・タイプのレコード・チェーンを更新する(ステップ473~474)。 $N \times k(AD)$ を M に設定する前に、 H_k を $N \times k(AD)$ で置換する。ステップ474の後、プロセスはステップ450へと戻る。 20

【0223】

テスト451で $k = 1$ になったら、図35のループは終了である。

【0224】

図34Cおよび図34Dは、図34Aおよび図34Bと等しい内容を有するテーブルを示し、ここでは $k > 1$ のビットマップ・セグメント HP_k が明示的に記憶されていない。層 n のコード化データ・コンテナ(図34C)は、図34Aに関して説明したものと同一であるが、 HP_n 列がない。 $k < n$ の各層 k コード化データ・コンテナ(図34D)は、図34Bに関して説明したものと同一であるが、層 $k+1$ の剰余を含む追加列 $R(k+1)$ がある。層 k コード化データ・コンテナへのレコードに剰余値 $R(k+1)$ が存在することは、記憶されていない上位層のビットマップ・セグメント $HP(k+1)$ の位置 $R(k+1)$ に「1」があることを意味する。 30

【0225】

図17の方式、すなわち、ディスク・アクセスを最少にするために、シソーラス項目ごとにコード化データ・ファイルを別個にする方式は、記憶してあるコード化データが層1、
 ・ ・ ・ , $n-1$ のランクを含まない場合にも適用することができる。層 n のランクおよびビットマップ・セグメントは、図17のように記憶することができる。下位層については、幾つかの選択肢がある。シソーラス・ワードおよびコード化層 $k < n$ ごとに1つのデータ・コンテナがあり、データ・チェーンが、上位層 $k+1$ に関するレコードで指示される。層 k のレコード・チェーンは、属性名 AT 、ワードまたはマクロワード値 W 、コード化層索引 k および層 $k+1$ のランク $NO(k+1)$ を含む名前を有する別個のファイルに切り離すこともできる。このようなファイル $AT\_W\_k\_NO(k+1)$ の各レコードは、層 $k+1$ の剰余 $R(k+1)$ 、およびランク $NO_k = NO(k+1) \times D(k+1) + R(k+1)$ で検出される層 k のビットマップ・セグメント HP_k を含む。 40

【0226】

問合せ基準の取扱い

$RDBMS$ の場合と同様、問合せは、構造的問合せ言語(SQL)に従って表現することができ、これは国際標準化機構(ISO)および米国規格協会($ANSI$)が標準として 50

採用している。

【0227】

問合せ処理手順の一般的な流れ図を図36に示す。

【0228】

SQLの「WHERE」文節に含まれる問合せ基準を、図36の段階190の要求木に変換する。問合せ基準を分析し、葉がSQL問合せで定義されたような個々の属性値の範囲に対応し、ノードがその葉から実行される論理演算に対応する木に従って構築する。葉は、SQL問合せの「BETWEEN文節」とも呼ばれる。SQL問合せで定義される個々の属性値は、1つのワードを対象とするBETWEEN文節である。

【0229】

例3：

このような木の例を、問合せの例示としてのケースの図37で示し、これは顧客Andreまたは顧客Maxが遭遇した事故に関し、 $500 \leq AA \leq 5000$ などの損害賠償金額を有する全データ・グラフを検索することを含む。この木は、破線で示した3つの葉を有し、これは問合せで定義されたBETWEEN文節に対応する。すなわち、顧客名属性については[Andre, Andre]および[Max, Max]、事故金額属性については[500, 5000]である。木は2つのノードも有し、一方は2つのCN基準間のOR演算、一方はAA基準でのAND演算用の根にある。

【0230】

木の分解は一意ではない。最小数のノードを有する方法を選択することが好ましい。

【0231】

次の段階191は、マクロワード・シソーラスを有する属性に関するBETWEEN文節を分析し、分割することによって、木の拡張を実行する。これは、ステップ190で獲得した木から、記憶してあるシソーラス・ワード、および問合せに使用する属性に関連付けられたマクロワード索引ファイルに関して実行される。BETWEEN文節で定義された各範囲の上下の境界を、属性に関連するシソーラスのワードと比較して、範囲から部分範囲への分解を検索し、これによってワードまたはマクロワード・シソーラス内で各部分範囲もBETWEEN文節として定義される。

【0232】

好ましい実施形態では、マクロワードを最大限に使用するため、分解を最適化する。この最適化では、可能な最小数のワードまたはマクロワードを保持し、BETWEEN文節で定義される範囲にマッピングされる部分範囲を形成する。システムは、インターバルに含まれる最高レベルのマクロワードを選択し、最小のワード・レベルを獲得するか、その範囲のワードがなくなるまで範囲の残りの部分で同じプロセスを繰り返す。

【0233】

段階191で生成された拡張木では、分割されているBETWEEN葉を、ORノードおよび部分範囲に関連付けられた葉で作成した部分木で置換する。この葉もBETWEEN文節の形式であり、問合せに関係あるシソーラス項目をカバーする。拡張木は、その後の処理のため、関係あるシソーラス項目の組合せを定義する。

【0234】

拡張木の全ての葉は、個々のワードまたはマクロワードの（部分）範囲と関連付けられる。このような範囲は、シソーラス索引ファイルのワードまたはマクロワード行IDの点で、その境界によって定義することができる。

【0235】

図38は、例3（図37）の木に対応する拡張木を示す。これは、図25～図27のシソーラス索引ファイルによって取得される。1ワードの範囲「CN=Andre」および「CN=Max」は分割されないが、単に、二分検索で取得したシソーラス索引ファイルのワードの行IDのCN_x=1および4で符号化される。図26および図27の事故金額シソーラスをさらに検索すると、範囲 $500 \leq AA \leq 5000$ が部分範囲に分割され、一方はワードAA_x=2および5、一方はマクロワードAA_3_x=1に関する。

10

20

30

40

50

【0236】

図39は、図36の段階191でBETWEEN文節を分割するための最適手順の流れ図を示す。(接続された)範囲は空値を含まないと仮定する(さもなければ、葉を最初に、ORノードで連結された2つの置換葉に分割することができ、一方の葉は個々のワード行ID、AT_x=0を有し、他方は上記の仮定を満足する)。

【0237】

BETWEEN文節で考察する属性ATが、整数レベル・パラメータ q ($1 \leq q \leq Q$)で索引付けられたマクロワード・シソーラスの数 $Q \geq 0$ を有し、レベル $q=0$ が個々のワード・シソーラスを指定することも仮定する。レベル q のシソーラスでは、接頭語長(例えば、ASCII文字の数)を $P(q)$ と表記し、 $P(0) > P(1) > \dots > P(Q)$ である。 $P(0)$ は個々のワード長である。図39～図40では、 x_{max} がシソーラス0の非空ワードの数を指定し、 $W_q(x)$ がレベル q シソーラスの行ID= x に記憶された(マクロ)ワードを指定し、 $[W]P(q)$ が、長さ $P(q)$ の接頭語($q \geq 1$)を維持するためにワード W を切り捨てることによって取得されたマクロワードを指定する。

【0238】

図39の手順の初期ステップ200で、プログラムはワード・シソーラス行IDの a および b を選択し、したがって $W_0(a)$ および $W_0(b)$ はそれぞれ、処理中の葉のために定義された範囲に含まれる最低および最高シソーラス・ワードである。整数 a および b は、範囲の境界に基づき、ワード・シソーラスの二分検索で容易に選択される。検索によって、範囲の対象にシソーラス・ワードがないことが判明した場合、手順は、葉の出力が空白のフラットファイル行IDリストとなることを指定して終了する。

【0239】

$W_0(a)$ がシソーラスの最低ワードである場合は(テスト201で $a=1$)、2進変数 XL をステップ202で $XL=0$ として初期化する。それ以外では、ステップ203で $XL=1$ として初期化する。 $W_0(b)$ がシソーラスの最高ワードである場合は(テスト204で $b=x_{max}$)、2進変数 XR をステップ205で $XR=0$ として初期化する。それ以外では、ステップ206で $XR=1$ として初期化する。以下のステップでは、値 XL (XR)=0は、考察中の範囲の下(上)限がマクロワードの境界と一致することを示す。レベル q シソーラスからのマクロワード境界と一致する場合は、 $1 \leq q' \leq q$ である任意のレベル q' のシソーラスでも真である。初期化201～206は、 $q=Q$ の場合に有効である。

【0240】

ステップ207で、プログラムは図40で示した流れ図を有するファンクションFUNCを起動する。このファンクションは、処理した葉の場所に挿入すべき部分木を記述するデータを戻す(ステップ208)。ファンクションFUNCは、図40Aのステップ210で実行を開始した時に入力された6つの引数を有する。すなわち、属性基準AT、シソーラス・レベル・パラメータ q (図39のステップ207でファンクションを最初に開始した時には $q=Q$)、問題となる範囲にある最低および最高ATワードのシソーラスの行IDの a および b 、および上記で定義した変数 XL および XR である。

【0241】

ステップ210の後、シソーラス・レベル・パラメータ q がゼロか否かを判断する(テスト211)。 $q > 0$ の場合は、 $W_q(a') = [W_0(a)]_{P(q)}$ および $W_q(b') = [W_0(b)]_{P(q)}$ となるように、ステップ212で2つのマクロワード・シソーラスの行ID a' および b' を選択する。これは、ワード $W_0(a)$ および $W_0(b)$ を切り捨てた後にレベル q シソーラスの簡単な二分検索によって実行される。

【0242】

その後のテスト213では、変数 XL を評価する。 $XL=1$ の場合は、テスト214で連続するワード $W_0(a-1)$ および $W_0(a)$ が同じレベル q のマクロワードを共用するか、すなわち、 $[W_0(a-1)]_{P(q)} = W_q(a')$ であるか判断する。イエスである場合は、ステップ215で整数 a' を1ユニットだけ増加させる。テスト214で

10

20

30

40

50

$W_0(a-1)]_{P(q)} < W_q(a')$ である場合は、ステップ216でXLの値を0に変更する。考察中の範囲の下限が、レベルqのマクロワードの境界と一致するからである。ステップ215または216の後、またはテスト213で $XL=0$ の場合は、変数XRを評価する(テスト217)。XR=1の場合は、テスト218で連続するワード $W_0(b)$ および $W_0(b+1)$ が同じレベルqのマクロワードを共用するか、すなわち、 $[W_0(b+1)]_{P(q)} = W_q(b')$ であるか判断する。イエスである場合は、ステップ219で整数 b' を1ユニットだけ減少させる。テスト218で $[W_0(b+1)]_{P(q)} > W_q(b')$ である場合は、ステップ216でXRの値を0に変更する。考察中の範囲の上限が、レベルqのマクロワードの境界と一致するからである。

【0243】

10

ステップ219または220の後、またはテスト217でXR=0の場合は、テスト221で変数 a' および b' を比較する。 $a' > b'$ である場合は、考察中の範囲がレベルqのマクロワードに及ばず、プログラムはステップ222でqを1ユニットだけ減少させ、ステップ211へと戻る。

【0244】

テスト221で $a' \leq b'$ である場合は、拡張した問合せ木に挿入するため、 $b' - a' + 1$ マクロワードの部分範囲を作成する(図40Bのステップ223)。この部分範囲は、 $AT\_P(q) \_x = a'$ から $AT\_P(q) \_x = b'$ まで、マクロワード行IDをカバーする。

【0245】

20

その後、ステップ224で再び変数XLを評価する。XL=1の場合は、ステップ223で作成した部分範囲の下に別の範囲を考慮しなければならない。ステップ225で、その下方範囲の上限の行ID、 b'' を決定する。対応するワード $W_0(b'')$ はATシソーラスの最上位であり、したがって $[W_0(b'')]_{P(q)} < W_q(a')$ である。次に、ステップ226でファンクション $FUNC(AT, q-1, a, b'', 1, 0)$ を再帰的に呼び出して、追加の下限を扱う。ステップ226の後、またはテスト224でXL=0の場合は、ステップ227で変数XRを再び評価する。XR=1の場合は、ステップ223で作成した部分範囲の上に、別の範囲を考慮しなければならない。ステップ228で、その上方範囲の下限の行ID、 a'' を決定する。対応するワード $W_0(a'')$ はATシソーラスの最下位であり、したがって $[W_0(a'')]_{P(q)} > W_q(b')$ である。次に、ステップ229でファンクション $FUNC(AT, q-1, a'', b, 0, 1)$ を再帰的に呼び出して、追加の下限を扱う。

30

【0246】

テスト211で $q=1$ である場合は、拡張した問合せ木に挿入するため、 $b-a+1$ ワードの部分範囲を作成する(ステップ230)。この部分範囲は、 $AT\_x = a$ から $AT\_x = b$ まで、個々のワード行IDをカバーする。

【0247】

ステップ229または230の後、またはテスト227でXR=0である場合は、ステップ231で、ステップ223または230で作成されたか、ステップ226および/または229で再帰的に呼び出されたファンクションで戻された、またはその両方の部分木を記述するデータを戻すことにより、ファンクションFUNCの実行を終了する。

40

【0248】

問合せ木を分析し、拡張する段階191が終了したら、図36の段階192で、最高コード化層nから開始して、拡張木を処理する。 $n > 1$ の場合は、図41のループで示すように、層kにて、kをnから1へと減少させて処理を連続的に実行する。

【0249】

図41のステップ240で、コード化層索引kを値nで初期化する。層kの処理は、ステップ241で開始し、拡張した問合せ木の根NDを、FNODEと命名されたファンクションを読み出す(ステップ242)ためのノードとして選択する。このファンクションへの入力、コード化層索引k、ノードNDおよびその子ノードを記述するパラメータ、お

50

よびビットマップ・ベクトル Res ($k = n$ で任意の方法にて初期化される) を備える。その出力は、 WZ と表記されるビットマップ・ベクトルである。層 1 では、出力ビットマップ・ベクトル WZ のビットの値が 1 であれば、 VDG (フラットファイル行 ID) が、ノード ND である根を有する木によって定義された問合せ基準と一致することを示す。層 $k > 1$ では、少なくとも 1 つの一致するフラットファイル ID を有する Δk 個のフラットファイル行 ID でできたグループの個々の層 $k - 1$ ランクを示す。各コード化層索引 k で、ファンクション $FNODE$ を再帰的に呼び出して、拡張した問合せ木の全ノードを処理する。

【0250】

ステップ 242 で呼び出したファンクションによって出力されたビットマップ・ベクトル WZ は、 $k > 1$ の場合、その後の層 $k - 1$ の処理で使用するため、ステップ 243 で層 k の問合せ結果 Res として保存する。その場合 (テスト 244) は、ステップ 245 で索引 k を 1 ユニットだけ減少させ、ステップ 241 から次の層処理を開始する。

【0251】

$k = 1$ の場合、 Res は、ステップ 246 で出力された所望のフラットファイル行 ID リストをビットマップで表示したものである。

【0252】

ファンクション $FNODE$ の流れ図を図 42 に示す。ビットマップ・ベクトル WZ は、個々では一続きのビット Dk 個のセグメントと見なされる。ベクトル WZ のランク N のセグメント (すなわち、 $N \geq 0$ の $(N + 1)$ 番目のセグメント) を $WZ[N]$ と表記する。ベクトル WZ のランク N のビット (すなわち、 $N \geq 0$ の $(N + 1)$ 番目のビット) を $WZ(N)$ と表記する。ファンクションの開始 (ステップ 248) 後、ビットマップ・ベクトル WZ を含む (ステップ 249) ために、 $RAM103$ に作業ゾーンを確保する。

【0253】

テスト 250 で、 ND が事前設定したノードを指示しているか最初に判断する。事前設定ノード (図 38 の例では図示せず) は、フラットファイル行 ID リストが既に決定されているノードである。通常、このリストは、以前の問合せの処理で一致するデータ・グラフ識別子 (ステップ 192 の出力) として作成されている。これは、このようなマッチング識別子リストの組合せでもよい。SQL 問合せが、以前の 1 つまたは複数の問合せ結果に言及する、例えば、応答を、以前の問合せに対する応答に含まれたレコードに制限している場合は、変換ステップ 190 で 1 つまたは複数の事前設定ノードを定義することができる。この機能は、データベースを対話モードで使用する場合に特に有用である。

【0254】

事前設定ノードについて以前に決定したフラットファイル行 ID リストは、 $RAM103$ に記憶するか、ハード・ドライブ 105 に保存することができる (後者の場合は、圧縮した形式で行うことが好ましい)。このリストは、 $1 \leq k \leq n$ の場合にビットマップ・ベクトルの形式で層 k の入力リストを提供するように、 n コード化層に従って符号化される。このような層 k のビットマップ・ベクトルは、テスト 250 でノード ND が事前設定されていると判明した場合、ステップ 251 で WZ としてロードされる。

【0255】

それ以外では、演算子ノードが葉ではなく演算子ノードを指示した場合 (テスト 252) は、ステップ 253 でその最初の子ノード $ND1$ を選択し、ステップ 254 でファンクション $FNODE$ を再帰的に呼び出し、ノード $ND1$ に対応するビットマップ・ベクトル $WZ1$ を取得する。次にステップ 255 で演算子ノード ND の第 2 の子ノード $ND2$ を選択し、ステップ 256 で再びファンクション $FNODE$ を呼び出して、ノード $ND2$ に対応するビットマップ・ベクトル $WZ2$ を取得する。

【0256】

ステップ 257 では、ビットマップ・ベクトル $WZ1$ を $WZ2$ とビットごとに組み合わせてビットマップ・ベクトル WZ を形成する。この組合せ

【0257】

10

20

30

40

50

【数 1 2】

(如何なる N に対しても $WZ(N) = WZ1(N) \otimes WZ(n)$)

は、例えば、AND、OR、Exclusive ORなどの演算のように、ノードNDのパラメータで記述されたブール演算子

【0 2 5 8】

【数 1 3】

$\otimes$

10

に従う。これは基本的にビットマップ・ベクトルの重ね合わせであり、両方のオペランド・ベクトルがRAM103に記憶されているので、非常に迅速に実行される。ステップ258では、作業ゾーンWZ1およびWZ2に割り当てられているRAMスペースを解放する。図42では、演算子ノードが2つの子ノードを有するケースのみを考察する。これは明らかに3つ以上のオペランドがあるケースにも拡張することができる。さらに、NOT演算のように、1つのオペランドの演算もあり、したがってファンクションFNODEを1回呼び出すだけでよい。

【0 2 5 9】

ノードNDが葉である場合（テスト252）は、初期化ステップ260で作業ゾーンWZの全ビットをゼロに設定する。また、シソーラス・ポインタxを、ノードNDについて定義されたBETWEEN範囲の第1行IDの値x1に初期化する。

20

【0 2 6 0】

ノードNDが、属性AT、およびシソーラスが「低密度」フォーマットで記憶されているマクロワード索引qに関係がある場合（テスト261）は、葉を図43に関して以下で説明するように処理し（ステップ262）、関連するビットマップ・ベクトルWZを取得する。シソーラス・フォーマットが「通常密度」である場合、処理はプログラムが（時間の経過順に）第1の層にあるか、すなわち、 $k=n$ であるか否か（テスト263）によって決定される。図44の処理は、 $k=n$ の場合に当てはまり（ステップ264）、 $k<n$ の場合は図45の処理が当てはまる（ステップ265）。

【0 2 6 1】

ステップ251、258、262、264または265の後、ステップ266でビットマップ・ベクトルWZを戻すことにより、ファンクションFNODEの実行を終了する。

30

【0 2 6 2】

低密度処理を説明するために、図43では、シソーラスの記憶機構がレコード・チェーンも利用すると仮定する。シソーラスが、図25～図27と同様の索引ファイル（行IDxに記憶されたワード索引はAT_WI(x)と表記される）と、ワード索引によってアドレス指定され、レコード・チェーンの第1のフラットファイル行IDのデータ・コンテナにアドレスAT_F(WI)を含む補助テーブルを有する。アドレスAD>0の各レコードで、このデータ・コンテナは、フラットファイル行ID値NO(AD)に加えて、レコード・チェーンの次アドレスNX(AD)へのポインタを含む次アドレス・フィールドを有する。チェーン末尾は、NX(AD)=0を有する。あるいは、低密度リストを、ワードごとに個々のファイルに記憶することができる（図17と同様）。

40

【0 2 6 3】

図43の低密度処理は、BETWEEN範囲のワードを連続的に取り扱うループを有する。反復するごとに、プログラムは最初にステップ270でワード索引WI=AT_WI(x)を、次にステップ271で頭部アドレスAD=AT_F(WI)を取得して、レコード・チェーンの走査を開始する。AD>0である場合（テスト272）は、レコード・チェーンに検査すべき少なくとも1つの項目が残っているので、ステップ273でフラットファイル行ID値NO(AD)および次アドレスNX(AD)をそれぞれ変数NおよびMとして読み出す。ステップ274で、

50

【0264】

【数14】

$$\Delta k = \prod_{k'=1}^{k-1} Dk' \quad (\Delta 1 = 1)$$

によるNのユークリッド除法を実行し、層 $k = 1$ の商（ランク） N' を取得する。 $k = 1$ の場合は $N' = N$ である。 $k > 1$ の場合、この演算274は、層 k' の除数 Dk' が $2^{\delta k'}$ （ $\delta k'$ は整数（ $1 \leq k' < k$ ））であれば、 N （剰余）の

【0265】

【数15】

$$\sum_{k'=1}^{k-1} \delta k'$$

最下位ビットを削除するだけである。次に、ビットマップ・ベクトル WZ のビット WZ （ N' ）に「1」を書き込む（ステップ275）。ステップ276で AD を次アドレス M で置換した後、テスト272に戻る。レコード・チェーンを完全に検査したら（テスト272で $AD = 0$ ）、現ワード x が $BETWEEN$ 範囲の最終ワード x_2 であるか判断する（テスト277）。 $x < x_2$ の場合は、ループの次の反復のために、ステップ278でシソーラス・ポインタ x を1ユニットだけ増加させる。テスト277で $x = x_2$ であればループが終了し、プログラムは図42のステップ266へと進む。

【0266】

図44の層 n の通常密度処理は、 $BETWEEN$ 範囲のワードまたはマクロワードを連続的に取り扱う同様のループを有するが、（記憶してある）コード化データの再計算は実行しない。反復ごとに、プログラムは最初にステップ280でワード索引 $WI = AT\_P(q) \_ WI(x)$ を取得し、次にステップ281で頭部アドレス $AD = AT\_P(q) \_ Fn(WI)$ を取得して、レコード・チェーンの走査を開始する。 $AD > 0$ である場合（テスト282）は、レコード・チェーンに検査すべき少なくとも1つの項目が残っているので、ステップ283で層 n のランク値 $NO_n(AD)$ 、次アドレス $NX_n(AD)$ および対応する層 n のビットマップ・セグメント $HP_n(AD)$ をそれぞれ変数 N 、 M および H として読み出す。次にビットマップ・セグメント H を、ブール OR 演算でビットマップ・ベクトル WZ のセグメント $WZ[N]$ に重畳させ（ステップ284）、ステップ285で AD を M で置換してからテスト282に戻る。レコード・チェーンを完全に検査したら（テスト282で $AD = 0$ ）、現ワード x が $BETWEEN$ 範囲の最終ワード x_2 であるか判断する（テスト286）。 $x < x_2$ の場合は、ループの次の反復のために、ステップ287でシソーラス・ポインタ x を1ユニットだけ増加させる。テスト286で $x = x_2$ であればループが終了し、プログラムは図42のステップ266へと進む。

【0267】

シソーラスを図25～図32に示すように構成したケースで、層 $k < n$ の通常密度処理を図45で詳細に示す。これは、 $NO_k(AD)$ が、 $BETWEEN$ 範囲のワードまたはマクロワードに関連する層 k のランク・リストに属していても、先行層 $k+1$ で取得したビットマップ・ベクトル Res のランク $NO_k(AD)$ のビットにゼロがあれば、ビットマップ・セグメント $HP_k(AD)$ へのアクセスは無駄である、ということを利用する。

【0268】

図45の手順は図44の手順と同様である。ステップ280～282と285～287は、 n を k で置換すれば同じである。しかし、レコード・チェーンを検査すべき場合（テスト282で $AD > 0$ ）は、ステップ290で層 k のランク値 $NO_k(AD)$ および次アドレス $NX_k(AD)$ のみを変数 N および M として読み出す。次に、テスト291で層 $k+1$ の結果ビットマップ Res のビット $Res(N)$ を評価する。 $Res(N) = 0$ の場合

は、ステップ285に直接ジャンプして、ランクNをフィルタリングし、除去する。それ以外 ($Res(N) = 1$) では、ステップ293でビットマップ・セグメント $HP_k(AD)$ を読み出してから、ステップ284へと進む。

【0269】

シソーラス項目のコード化データを構成すると、図44および図45のループが一般に、連続するデータ・コンテナのレコードの連続的読み取りを意味することが分かる (ステップ283および290)。何故なら、索引WIの各ワードが、データ・コンテナの連続するアドレスADに記憶されたコード化データ、およびBETWEEN範囲の大部分の連続ワードを有するからである。したがって、これらのループは、コンピュータのキャッシュ・メモリでデータ・コンテナレコードのブロックをロードし、それによって必要なディスク・アクセス回数を減少させることによって、効率的に実行することができる。同じ考慮事項が、図43のステップ273で読み出す低密度データ $NO(AD)$ および $NX(AD)$ にも当てはまる。

10

【0270】

図46に示した層 $k < n$ の通常密度処理で、さらに改良が得られ、これは2つの連続ループで構成される。シソーラス・ポインタxで索引付けられた第1のループは、第2のループでビットマップ・セグメントを取り扱うために使用する TNO と表記された一時的ランク・テーブルを決定するためのものである。テーブル TNO は幾つかのアドレスを有し、これは少なくとも、現シソーラスの層kコード化データ (AT, q) を記憶しているデータ・コンテナのアドレス AD_{max} の数と等しい。ランク・テーブル TNO のアドレスADの各項目 $TNO(AD)$ は、ビットマップ・セグメント $HP_k(AD)$ へのアクセスが有用な場合は、ランク $NO_k(AD)$ を表す整数、それ以外ではデフォルト値 (-1) を含むためのものである。

20

【0271】

初期化ステップ279では、ランク・テーブル TNO の全項目をデフォルト値 -1 に設定する。第1のループは図45のループと同様である。テスト291で $Res(N) = 1$ である場合は、ステップ295でランクNをアドレスADでテーブル TNO に書き込んでから、ステップ285でADをMで置換する。

【0272】

第1のループが終了したら (テスト286で $x = x2$ になったら)、プログラムは第2のループへと進み、これはステップ301にて $AD = 1$ で初期化される。第2のループを反復するごとに、アドレスADでのランク・テーブル TNO の内容Nをステップ302で読み出し、テスト303でデフォルト値と比較する。Nが有効ランク値 ($\neq -1$) である場合は、ビットマップ・セグメント $HP_k(AD)$ を読み出し (ステップ304)、ビットごとにブールOR演算でビットマップ・ベクトルWZのセグメントWZ[N]に重畳させる (ステップ305)。 $AD < AD_{max}$ である場合 (テスト306) は、ステップ307でランク・テーブルのアドレスADを1ユニットだけ増加させてから、ステップ302へと戻る。テスト306で $AD = AD_{max}$ になった場合は、第2のループが終了し、プログラムは図42のステップ266へと進む。

30

【0273】

読み出す価値がないビットマップ・セグメント $HP_k(AD)$ をフィルタリングして除去する以外に、図46で示した手順は、ランク・テーブル TNO のおかげで、アドレスADに基づいて層kのビットマップ・セグメント・データを含むファイルの読み出し作業をグループ分けする (第2のループのステップ304)。このようなグループ分けは、ワード単位ばかりでなく、BETWEEN範囲の全ワードでも実行され、最終的に第2のループで HP_k ファイルを読み出す場合に、ランク値がテーブル TNO に書き込まれているワード間には区別がなくなる。これは、 HP_k ファイルのブロックごとのアクセスを最大限に利用し、非常に大きい利点を提供する。何故なら、下位層、特に層1は最大の HP_k を意味し、ここで読み出し作業数が最高になるからである。

40

【0274】

50

図 47 は、コード化データ・コンテナが図 25～図 30 および図 34A～図 34B で示したように記憶されている場合に、図 45 の手順を適用する方法を示す。ループは同様の構造を有する。しかし、コード化データを最高層 n からアクセスするので、ステップ 281 で読み出すアドレス AD は、層 n のデータ・コンテナにあるレコード・チェーンの頭部アドレス $AT\_P(q)\_Fn(WI)$ であり、ステップ 282 で $AD > 0$ である場合は、ステップ 296 で変数 N および M として読み出されるランク値 $NO_n(AD)$ および次アドレス $NX_n(AD)$ は、層 n とも関係する。ステップ 296 の後、ステップ 297 でフィルタリング・ファンクション $FILT$ を呼び出してから、ステップ 285 で AD を M で置換する。

【0275】

10

このファンクション $FILT$ の流れ図を図 48 に示す。ステップ 500 で実行を開始した時に入力されたその引数は（図 47～図 48 で暗黙的な属性名およびマクロワード・レベル以外に）、以下の通りである。

- 図 41 のステップ 242 で呼び出したファンクション $FNODE$ の第 1 の引数に対応する第 1 のコード化層索引 k ；
- 図 47 のステップ 297 でファンクション $FILT$ を呼び出した場合に、 $k' = n$ である、第 2 のコード化層索引 $k' > k$ ；
- $k' - k$ ビットマップ・ベクトル $Res_{k+1}, Res_{k+2}, \dots, Res_k$ 、ここで、 Res_{k+1} は層 $k+1$ の問合せ結果 Res である。 $k' > k+1$ の場合、 $Res_{k+2}, \dots, Res_k$ は、図 41 のステップ 243 で上位層にて Res を符号化して取得されたビットマップ・ベクトルである；
- 層 k' のランク N で、図 47 のステップ 297 でファンクション $FILT$ を呼び出した場合は $N = NO_n(AD)$ である；
- 層 k' のデータ・コンテナにあり、対応するレコード・アドレス AD ；および
- 計算中のビットマップ・ベクトル WZ である。

20

【0276】

テスト 501 では、ビットマップ・ベクトル Res_k の $(N+1)$ 番目のセグメントがゼロになっているだけか判断する。イエスの場合は、層 k' のランク N に関するコード化データをさらに読み出す必要はなく、したがってステップ 502 でビットマップ・ベクトル WZ を戻して、ファンクションが終了する。

30

【0277】

テスト 501 でセグメント $Res_k[N]$ が少なくとも 1 つの「1」を有する場合は、ステップ 503 でビットマップ・セグメント $HP_{k'}(AD)$ をセグメント変数 H として読み出し、テスト 504 で交差セグメント $H \text{ AND } Res_k[N]$ を評価する。この交差セグメントがゼロになっているだけの場合は、同様にコード化データをさらに読み出すのは無駄であり、プログラムは直接ステップ 502 へと進む。

【0278】

テスト 504 でセグメント $H \text{ AND } Res_k[N]$ が少なくとも 1 つの「1」を有していると判明した場合は、下位層のレコード・チェーンへと入る必要がある。ステップ 505 でその頭部アドレス $F(k'-1)(AD)$ を変数 AD' として読み出し、その一方でランク k' の剰余 R を 0 に初期化し、層 $k'-1$ のランク N' を $N \times D_{k'}$ に初期化する。ビット $H(R)$ を連続的に検査するループ（テスト 506）でビットマップ・セグメント $H = HP_{k'}(AD)$ を走査し、ランク $N' = N \times D_{k'} + R$ を考慮すべきか確認する。 $H(R) = 0$ の場合は、ランク N' が現シソーラス項目の層 k' のコード化データに入っていないので、これは無視される。すなわち、ステップ 507 で R を 1 ユニットだけ増加させ、新しい R がそれでも $D_{k'}$ より小さい場合（テスト 508）は、ステップ 509 で N' も 1 ユニットだけ増加させてから、テスト 506 から次の反復へと進む。

40

【0279】

テスト 506 で $H(R) = 1$ である場合は、テスト 510 でベクトル Res_k の $(N'+1)$ 番目のビットを検査し、層 $k'-1$ のランク N' を、上位層の処理でフィルタリン

50

グして除去されているか判断する。除去されている場合 ($Res_k, (N') = 0$)、プログラムは、ステップ 511 で AD' を $NX(k' - 1)(AD')$ で置換することにより、層 $k' - 1$ のレコード・チェーンの次位置にジャンプする。ステップ 511 の後、プログラムは上述したステップ 507 へと進む。

【0280】

テスト 510 で $Res_k, (N') = 1$ である場合、処理は、コード化層 k' が k のすぐ上か否か (テスト 512) によって決定される。 $k' = k + 1$ の場合は、ビットマップ・セグメント $HP_k(AD')$ を読み出して (ステップ 513)、ビットごとにブール OR 演算でビットマップ・ベクトル WZ のセグメント $WZ[N']$ に重畳させる (ステップ 514)。テスト 512 で $k' > k + 1$ である場合は、ステップ 515 にて引数 k 、 $k' - 1$ 、 $Res_{k+1}, \dots, Res_{k' - 1}, N', AD'$ および WZ で再帰的ファンクション $FILT$ を呼び出す。ステップ 514 または 515 の後、プログラムは上述したステップ 511 へと進む。

【0281】

テスト 508 で $R = D_{k'}$ になったら、ビットマップ・セグメント $H = HP_{k'}(AD)$ の走査は終了である。次に、ステップ 502 で更新したビットマップ・ベクトル WZ を戻す。

【0282】

コード化データ・コンテナを図 34C～図 34D で示すように構成すると、ループ 505 から 509 の層 k' ビットマップ・セグメントの走査が、層 $k' - 1$ のコード化データ・コンテナのレコード・チェーンにある層 k' の剰余の走査で置換される。

【0283】

図 47～図 48 の手順は、下位層のレコード・チェーンが厳密に必要な場合しかアクセスされないという利点を有する。特に、図 45 のループ 282 から 285 は、現シソーラス項目に関する全ての層 k ランクを読み出す必要がある (ステップ 290) が、 $k + 1$ の処理から一部のランクが無視される (テスト 291 で $Res(N) = 0$) ことが既に知られていることが分かる。図 47～図 48 でこれが生じると、ハード・ドライブでランク N を読み出さない (記憶もしない)。この利点が非常に重要であるのは、下位層、特に層 1 が最大のコード化データ・コンテナを有し、したがって多くの無駄な読み出し作業が回避されるからである。

【0284】

図 46 によるランク・テーブル TNO の使用は、図 47～図 48 の手順と非常に互換性があることが分かる。図 46 の第 1 のループ 280 から 287 は、単に図 47 のそれと置換され、図 48 のステップ 513 から 514 は、 N' を $TNO(AD')$ に書き込むことで置換される。

【0285】

図 43～図 47 のループは、シソーラスの $BETWEEN$ 範囲を対象とするばかりでなく、同じデータ・コンテナに記憶されたコード化データを有し、OR タイプの演算と組み合わせられるワードおよび／またはマクロワードも対象とすることが分かる。 $x = x_1$ から $x = x_2$ までループを実行する代わりに、このようなワードまたはマクロワードそれぞれで反復を実行する。

【0286】

例えば、任意の属性のワードおよびマクロワード・シソーラスが同じデータ・コンテナを共用している場合、ループは、属性の全ての関連値、すなわち、図 36 の段階 191 で問合せ木の対応するノードと置換された部分木で、1 回しか実行しなくてよい。

【0287】

また、このようなワードおよび／またはマクロワードが異なるシソーラスに属する可能性がある (問合せ木の OR ノードの適切なラベリングを必要とする)。例えば、問合せが、特定の顧客が遭遇した事故、または所定の値を上回る損害賠償金額を有する事故を目的とし、顧客および事故金額シソーラスが (図 31～図 32 のように) 同じデータ・コンテナ

を共用している場合は、図 4 6 の同じ第 1 のループ内で顧客および事故金額属性を検査し、TNO テーブルを 1 回だけ走査して、関連する全ての HP 1 セグメントを検索することができる。

【0288】

しかし、前述したようにシソーラスおよびマクロワード・レベルごとに 1 つのデータ・コンテナを有することが好ましい。その利点は、図 4 6 の手順で使用するランク・テーブル TNO のサイズを減少させることである。

【0289】

拡張した問合せ木の葉を符号化する場合は、シソーラス行 ID x ではなくワード索引 AT__P (q) __W I (x) の使用が可能であることも分かる。次に、拡張した問合せ木の葉ごとにワード索引のリストを符号化する。したがって、木の拡張手順 191 は、シソーラス・ワード索引ファイルに関連して実行されるが、段階 192 の処理では使用されず、これはワード索引によってアドレス指定されたレコード・チェーンの頭部を直接呼び出す。これは、(図 2 5 ~ 図 2 7 とは異なり) ワード索引がシソーラスの行 ID と一致していない場合に有用であり、これはデータベースが生きているので、普通に生じることである。

【0290】

図 1 7 のように、シソーラス・ワードごとに別個のコード化データ・ファイルを使用する好ましいケースでは、ステップ 264 の層 n の処理が、図 4 4 に示す処理と同様である。ループは、共通のデータ・コンテナ内で (ループ索引 AD で) 実行されるのではなく、個々のコード化データ・ファイル AT__P (q) __W__N O k および AT__P (q) __W__H P k で (図 3 3 のようにループ索引 i で) 実行される。シソーラスをソートせずに、最適なディスク・アクセスが確保される。ステップ 265 の層 k < n 処理は、図 4 6 と異なり、2 つのループを必要としない。図 4 9 の通りでよい。

【0291】

図 4 9 に示す手順の第 1 のステップ 310 は、現シソーラスにあるランク x のワードの値 AT__P (q) (x) をワード変数 W に割り当て、ループ索引 i を起動してゼロにすることにある。i が、シソーラス AT、マクロワード・レベル q およびワード W に関するコード化データ・ファイルにある層 k レコードの総数 i m a x (AT、q、W、k) より小さい限り (テスト 311)、ステップ 312 から 315 を実行する。ステップ 312 では、ランク AT__P (q) __W__N O k (i) を整数変数 N に割り当てる。これらのランク・データをブロックごとに読み出して、ディスク・アクセスを最少にする。その後のテスト 313 で、層 k + 1 の結果ビットマップ R e s のビット R e s (N) を評価する。R e s (N) = 1 である場合は、ステップ 314 でビットマップ・セグメント AT__P (q) __W__H P k (i) を読み出し、ステップ 315 でブール OR 演算によってビットマップ・ベクトル WZ のセグメント WZ [N] に重畳させ、それによって AT__P (q) __W__H P k (i) に「1」があれば、それに対応する位置で WZ [N] に書き込み、AT__P (q) __W__H P k (i) に「0」があれば、WZ [N] の対応するビットを元のままにしておく。ビットマップ・セグメント・データ AT__P (q) __W__H P k (i) は、ブロックごとにも読み出す。ステップ 315 の後、またはテスト 313 で R e s (N) = 0 の場合に実行されるステップ 316 では、ループ索引 i を 1 ユニットだけ増加させてから、テスト 311 へと戻る。関連するコード化データを完全に検査したら (テスト 311 で i = i m a x (AT、q、W、k))、現ワード x が B E T W E E N 範囲の最終ワード x 2 であるか判断する (テスト 317)。x < x 2 の場合は、ステップ 318 でシソーラス・ポインタ x を 1 ユニットだけ増加させてから、ループの次の反復のため、ステップ 310 へと戻る。テスト 317 で x = x 2 であれば、ループが終了している。

【0292】

図 5 0 は、図 2 5 ~ 図 2 7 の右部分で示すように、シソーラス指標レジスタでフラットファイル行 ID リストのコード化フォーマットが指定されているケースで、(テスト 252 がポジティブの場合に) 図 4 2 の葉処理を実行する代替方法を示す。

【0293】

初期化ステップ 260 A は図 42 のステップ 260 と同様であるが、リンク・テーブル TNO が同時にデフォルト値に初期化される。ステップ 280 A では、AT レベル q シソーラス指標レジスタからワード索引 $WI = AT\_P(q) \_ WI(x)$ および対応するフォーマット $F = AT\_P(q) \_ FORMAT(x)$ を読み出す。F が「低密度」を指定している場合 (テスト 261 A) は、図 43 で示したループ 271 から 276 をステップ 262 A で実行する。そうでない (F が n コード化層で「通常密度」を指定する) 場合は、ステップ 281 A で頭部アドレス $AD = AT\_P(q) \_ Fk(WI)$ を読み出して、レコード・チェーンの走査を開始する。第 1 のコード化層 $k = n$ (テスト 263 A) で作業中の場合は、図 44 で示したループ 282 から 285 をステップ 264 A で実行する。それ以外では、図 46 の第 1 のループ 282 から 285 をステップ 265 A で実行する。ステップ 262 A、264 A または 265 A の後、テスト 286 A で現シソーラス・ポイント x を BETWEEN 範囲の上限 $x2$ と比較し、ステップ 2870 A で増加させてから、 $x < x2$ である場合はステップ 280 A へと戻る。テスト 286 A で $x = x2$ である場合は、図 42 のステップ 266 で戻されたビットマップ・ベクトル WZ を完成するために、ステップ 301 A でテーブル TNO を利用し、これは図 46 の第 2 のループ 301 ~ 306 と同じである。

10

【0294】

問合せの出力

SQ L 問合せは、さらに、問合せ基準に一致したデータを応答内で表す方法を指定する。したがって、問合せ処理の次の段階 193 (図 36) は、段階 194 で表示するための結果の準備である。

20

【0295】

通常、問合せは、表示する応答に含めねばならない値を有する属性のリストを定義する (SQ L 問合せの「SELECT」および「FROM」文節で、FROM が関連するデータ・テーブルを指定し、SELECT がそのテーブルの関連列を指定する)。

【0296】

図 9 に示すタイプのリンク・テーブルを記憶する場合、リストされた属性に対応するそのリンク・テーブルの列を、一致する行で読み出し、関連するデータ・テーブルを指示するリンクを取得するため、図 41 のステップ 246 でビットマップ・ベクトル Res 出力で識別する。

30

【0297】

もう 1 つの可能な方法は、このような属性に関するシソーラスを走査し、結果ビットマップ・ベクトル Res とシソーラスの符号化した各ビットマップ・ベクトルとの間でビットごとにブール AND を計算する。これらのベクトル間にヒットがある (AND 出力ベクトルに「1」がある) ごとに、対応するシソーラス・ワードが表示されるか他の方法で処理される。これによって、リンクまたはデータ・テーブルを一切使用せずに、応答の属性値を検索することができる。

【0298】

AND 演算は、層 1 で直接実行することができる。これは、 $k = n$ から $k = 1$ まで層索引を減少させることにより、前述したように実行することもできる。それには、層 1 のビットマップ・ベクトル Res から計算できる層 k の結果が必要である。後者の選択肢は、多層 VDG 圧縮方式を利用することによりディスク・アクセスを最適化する。

40

【0299】

このような走査は、マクロワード・シソーラスを利用することにより加速してもよい。属性の最高レベル・シソーラスを最初に走査し、マクロワードでヒットが観察された場合のみ、所与のマクロワードでカバーされた下位レベル・シソーラスを走査する。

【0300】

図 51 は、対応するマクロワードおよび／またはワード・シソーラスを走査することにより、所与の属性 AT に応答して含むべき値にアクセスするのに適した手順を示し、これはマクロワード・グルーピングおよび VDG 圧縮方式の両方を十分に利用する。

50

【0301】

前述したように、属性ATは、個々の接頭語長 $P(q)$ を有し、 $P(0) > P(1) > \dots > P(Q)$ であるレベル・パラメータ q ($0 \leq q \leq Q$)で索引付けされた $Q+1 \geq 1$ 個のシソーラスを有し、レベル・パラメータ $q=0$ が個々のワード・シソーラスを指定し、その接頭語長が、属性のワード長に対応するものとする。図45の表記では、以下の通りになる。

ー QA は、問合せの結果に予想される正確さを示す $0 \leq QA \leq Q$ の整数であり、 QA は、正確さを最大にするには0に設定する。

ー シソーラス・ポインタ x_q は、レベル q のATシソーラス指標レジスタにある行IDである。

ー $q \geq QA$ の場合、 $WZ1_q$ は、問合せ基準に一致し、レベル q のシソーラス・ワード x_q に関して検査すべきデータ・グラフ識別子の層 q の目標リストを表すビットマップ・ベクトルである。初期化ステップ320では、図41のステップ246で出力される結果ビットマップ・ベクトル Res をベクトル $WZ1_Q$ に割り当て、これは問合せ基準に一致するフラットファイル行IDを表す。

ー $k > 1$ の場合、 WZk_q はビットマップ・ベクトルを指定し、ここでランク N の各ビット（すなわち、 $(N+1)$ 番目のビット）は、VDG圧縮方式 ($0 \leq q \leq Q$)に従って、 $WZ(k-1)_q$ の $D(k-1)$ ビットの $(N+1)$ 番目のセグメントが少なくとも1つの「1」を含むか示す。 WZk_q は、 $QA \leq q \leq Q$ および $1 \leq k \leq n$ の場合に層 k およびレベル q フィルタリング・リストと呼ばれる。作業ゾーンは、ハード・ドライブに記憶する必要があるビットマップ・ベクトル WZk_q を含むため、RAM103に確保される。

【0302】

初期化ステップ320では、最高レベル・シソーラスの走査を開始するため、索引 q および x_Q を $q=Q$ および $x_Q=0$ に設定する。変換ステップ321では、ビットマップ・ベクトル $WZ1_q$ を処理して、対応する上位層ベクトル WZk_Q ($1 < k \leq n$)を提供する。

【0303】

コード化層索引 k をステップ322で n に設定し、ファンクションFINTERを呼び出して（ステップ323）、シソーラス項目 x_q の層 k コード化データで表示された整数リストとビットマップ・ベクトル WZk_q で表示されたフィルタリング・リストとの間の交差を判断する。このファンクションへの入力は、コード化層索引 k 、（マクロ）ワード・シソーラス・レベル q 、（マクロ）ワード索引 $x=x_q$ 、およびビットマップ・ベクトル $WZ=WZk_q$ を備える。その出力は、同じ次元を有し、整数リスト交差を表す WX と表記された別のビットマップ・ベクトルである。

【0304】

ステップ323で呼び出したファンクションFINTERが出力したビットマップ・ベクトル WX をステップ324でテストして、そのビットの少なくとも1つが「1」であるか判断する。ノーの場合は、 x_q によって指示された（マクロ）ワードが、その問合せに関連する属性値をまったくカバーしないので、ステップ325でシソーラス・ポインタ x_q を1ユニットだけ増加させて、プログラムはステップ322へと戻り、レベル q シソーラスの次の（マクロ）ワードを検査する。

【0305】

ビットマップ・ベクトル WX が少なくとも1つの「1」を有し、 $k > 1$ であれば（その後のテスト326）、ステップ327で層索引 k を1ユニットだけ減少させる。次にステップ323から次の層処理を開始する。

【0306】

テスト326で $k=1$ の場合は、 $WX \neq 0$ が、フラットファイル行IDのリストのビットマップ表示であり、これは結果ビットマップ・ベクトル Res と現（マクロ）ワード x_q のコード化データとの両方で表される。

10

20

30

40

50

【0307】

$q > Q_A$ の場合 (テスト 330) は、ステップ 331 でこのビットマップ・ベクトル WX を $WZ_{1_{q-1}}$ として保存する。次に、下位レベル・シソーラスにあるマクロワード x_q の第 1 の「子」の行 ID 、 $AT\_P(q)\_FW(x_q)$ をレベル q シソーラスで読み出し、シソーラス・ポインタ x_{q-1} の開始値として割り当てる (ステップ 332)。次に、ステップ 333 でシソーラス・レベル q を 1 ユニットだけ減少させ、ステップ 321 から下位レベル処理を開始する。

【0308】

テスト 330 で $q = Q_A$ の場合は、 x_{Q_A} によって指示されたワード ($Q_A = 0$ の場合) または x_{Q_A} によって指示されたマクロワードがカバーするワード ($Q_A > 0$ の場合) が、問合せ基準に一致するデータ・グラフの属性値である。ステップ 335 で、このワードまたはそのシソーラス・ポインタ x_{Q_A} および対応するビットマップ・ベクトル WX に基づいて、特定の措置を執る。後者のベクトル識別子は、 AT 列に (マクロ) ワード x_{Q_A} を含み、問合せ基準を満足するフラットファイルの行を識別する。措置のタイプは、 SQL 問合せによって決定される。可能な様々な方法について、さらに説明する。

【0309】

ステップ 335 の後、行為レベル・ビットマップ・ベクトル WZ_{1_q} を更新して、 WX と同じ位置に「1」があれば、それを除去する。このような「1」は、 x_{Q_A} によって指示されたワードを有するデータ・グラフ ($Q_A = 1$ の場合)、または x_{Q_A} によって指示されたマクロワードがカバーするワード ($Q_A > 0$ の場合) を属性 AT の値として表し、したがってこれにヒットする他のワードはないので、除去することができる。更新を初期化するには、ステップ 336 で索引 q を Q と等しくする。ステップ 337 で、ブール演算 $WZ_{1_q} \text{ AND NOT } WX$ をビットごとに実行すると、結果は更新済みの WZ_{1_q} になる。結果のビットマップ・ベクトル WZ_{1_q} に少なくとも 1 つの「1」が残っていたら (テスト 338)、ステップ 339 でシソーラス・レベル索引 q を 1 ユニットだけ減少させ、ステップ 337 を繰り返す。

【0310】

WZ_{1_q} がテスト 338 でゼロのみで構成されている場合は、下位レベルで更新を継続する必要がない。 $q < Q$ なら (テスト 340)、 x_q によって指示された (マクロ) ワードが、その問合せに関連する属性値をもうカバーしていない。したがってステップ 341 でシソーラス・ポインタ x_q を増加させ、プログラムはステップ 321 へと戻り、レベル q シソーラスの次の (マクロ) ワードを検査する。

【0311】

テスト 340 で $q = Q$ になると、属性 AT のシソーラスの走査が終了する。

【0312】

ステップ 323 で呼び出されたファンクション $FINTE R$ は、シソーラスを図 25 ～ 図 32 で示したように記憶している場合、図 52 に示した流れ図に従ってもよい。これはステップ 350 で上述した入力引数 k 、 q 、 $x (= x_q)$ および $WZ (= WZ_{k_q})$ をロードして開始する。ステップ 351 で、ビットマップ・ベクトル WX をゼロで初期化する。プログラムは最初にステップ 352 でワード索引 $WI = AT\_P(q)\_WI(x)$ を取得し、次にステップ 353 で頭部アドレス $AD = AT\_P(q)\_Fk(WI)$ を取得して、データ・コンテナにある関連レコード・チェーンの走査を開始する。

【0313】

属性 AT のレベル q シソーラス項目が、「低密度」フォーマットで記憶されている場合 (テスト 354)、処理は、図 53 に関して以下で説明するようになり (ステップ 355)、交差ベクトル WX を取得する。フォーマットが「通常密度」である場合、処理は、プログラムが第 1 の層にあるか、すなわち、 $k = n$ であるかによって (テスト 356) 決定される。図 54 の処理は、 $k = n$ である場合 (ステップ 357) に当てはまり、 $k < n$ の場合は図 55 の処理が当てはまる (ステップ 358)。ステップ 355、357 または 358 の後、ファンクション $FINTE R$ の実行は、ステップ 359 でビットマップ・ベクトル

ルWXを戻して終了する。

【0314】

図53の低密度処理は、テスト360でアドレスADをチェーン終了値(0)と比較することによって各反復を開始するループを有する。AD>0の場合は、レコード・チェーンに検査すべき項目が少なくとも1つ残っているので、ステップ361でフラットファイル行ID値NO(AD)および次アドレスNX(AD)を変数NおよびMとして読み出す。ステップ362でΔkによるNのユークリッド除法を実行し、層k-1の商(ランク)N'を取得する。その後のテスト363でWZ(N')=1である場合は、ビットマップ・ベクトルWXのビットWX(N')に「1」を書き込む(ステップ364)。ステップ364の後、またはテスト363でWZ(N')=0の場合は、ステップ365でADを変数Mで置換してから、テスト360へと戻る。レコード・チェーンを完全に検査したら(テスト360でAD=0)、現(マクロ)ワードの低密度処理が終了し、プログラムは図52のステップ359へと進む。 10

【0315】

図54の層nの通常密度処理は、各反復がステップ370でアドレスADをチェーン終了値(0)と比較することにより開始する同様のループを有する。AD>0の場合は、ステップ371で層nのランク値NO_n(AD)および次アドレスNX_n(AD)をそれぞれ変数NおよびMとして読み出す。ビットマップ・ベクトルWZのランクNのセグメントが少なくとも1つの「1」を有する(その後のテスト372でWZ[N]≠0)場合は、ビットマップ・セグメントHP_n(AD)を読み出し(ステップ373)、ビットごとのブールAND演算でビットマップ・セグメントWZ[N]と組み合わせて、ビットマップ・ベクトルWXのセグメントWX[N]を提供する(ステップ374)。ステップ374の後、またはテスト372でWZ[N]=0の場合は、ステップ375でADを変数Mで置換してから、テスト370へと戻る。レコード・チェーンを完全に検査したら(テスト370でAD=0)、現(マクロ)ワードの層nの通常密度処理が終了し、プログラムは図52のステップ359へと進む。 20

【0316】

層k<nの通常密度処理は、2つの連続ループ(図55)で実行すると有利である。第1のループは、一時的ランク・テーブルTNOを決定するためのものであり、これは図46に関して前述した手順と同様に、第2のループのビットマップ・セグメントを取り扱うために使用される。テーブルTNOは幾つかのアドレスを有し、これは少なくとも、現シソーラス(AT、q)の層kのコード化データが記憶されているデータ・コンテナのアドレスAD_{max}と等しい。ランク・テーブルTNOのアドレスADの各項目TNO(AD)は、ビットマップ・セグメントHP_k(AD)にアクセスするのに有用な場合はランクNO_k(AD)を表す整数、それ以外の場合はデフォルト値(-1)を含むためのものである。このようなアクセスは、NO_k(AD)が、現(マクロ)ワードx_qに関連付けられた層kのランク・リストに属していない場合、またはビットマップ・ベクトルWZ=WZ_{k_q+1}のランクNO_k(AD)のセグメントにゼロしかない場合は無駄である。 30

【0317】

初期化ステップ380では、ランク・テーブルTNOの全項目をデフォルト値-1に設定する。第1のループの各反復は、ステップ381で、アドレスADをチェーン終了値(0)と比較して開始する。AD>0の場合は、ステップ382で層kのランク値NO_k(AD)および次アドレスNX_k(AD)をそれぞれ変数NおよびMとして読み出す。テスト383で、ビットマップ・ベクトルWZのランクNのセグメントWZ[N]を検査する。そのセグメントWZ[N]が少なくとも1つの「1」を有する(テスト383でWZ[N]≠0)の場合は、ステップ384でランクNをアドレスADにてテーブルTNOに書き込んでから、ステップ385でADをMで置換し、テスト381へと戻って、チェーンの次レコードを検査する。それ以外(WZ[N]=0)の場合は、ステップ385へと直接ジャンプして、ランクNをフィルタリングして除去する。 40

【0318】

レコード・チェーンを完全に検査したら（テスト 381 で $AD = 0$ ）、第 1 のループの終了である。次にプログラムは第 2 のループ 386 ～ 391 へと進む。第 2 のループの各反復では、ステップ 386 で AD を増加させた後にステップ 387 で読み出したアドレス AD でのランク・テーブル TNO の内容 N を、テスト 388 でデフォルト値と比較する。 N が有効ランク値（ $\neq -1$ ）なら、ビットマップ・セグメント $HPk(AD)$ を読み出して（ステップ 389）、ビットごとのブール AND 演算でビットマップ・セグメント $WZ[N]$ を結合し、ビットマップ・ベクトル WX のランク N のセグメント $WX[N]$ を提供する（ステップ 390）。 $AD < AD_{max}$ である場合（テスト 391）は、次の反復の開始時にステップ 386 でランク・テーブル・アドレス AD を 1 ユニットだけ増加させる。テスト 391 で $AD = AD_{max}$ であると第 2 のループが終了し、プログラムは図 52 のステップ 359 へと進む。

【0319】

図 51 ～ 図 55 に関して説明したようなシソーラスの走査は、以下のように幾つかの重大な利点を有する。

- 元のデータ・テーブルにアクセスする必要がある。したがって、メモリにデータ・テーブルを維持することが強制されない。記憶してある場合でも、ODBC などの比較的低いソフトウェア・インタフェースでアクセス可能であることが多い。走査方法は、このインタフェースを迂回すると有利である。
- コード化データ・コンテナのレコード・グルーピングを利用するので、ディスク・アクセスの面で非常に効率的である。図 53 ～ 図 55 の手順は、それぞれディスク・アクセスに関する図 43、図 44 および図 46 と同様であり、この点で上述した利点を提供する。

- 図 51 の手順も、ビットマップ・ベクトル $WZ1_q$ の更新（ループ 336 ～ 339）によって達成されるフィルタリングのせいで、非常に効率的である。このフィルタリングは、各フラットファイル行が属性ごとに一意の値（場合によっては空）を有することを利用する。最下層でヒットがないようなその後のシソーラス・ワードおよびマクロワードに属するコード化データを読み出すための大量の無駄な作業（何故なら、上位層でのヒットは、既に考察したシソーラス・ワードに対応するフラットファイル行 ID によるものだからである）を回避する。

【0320】

図 56 は、コード化データ・コンテナを図 25 ～ 図 30 および図 34A ～ 図 34B で示すように記憶した場合に、図 51 の手順を適用する方法を示す。上述したファンクション $FINTER$ を、図 57 で示した再帰的ファンクション $FFILT$ で置換する。したがって、図 52 のように（ $k = n$ で）ステップ 351 から 353 を実行した後、ループ 322 から 327 をループ 590 から 593 で置換する。結果の交差ビットマップ WX がゼロのみで構成されている場合（テスト 324）は、ステップ 325 で x_q を増加させてから、現レベル q のシソーラス範囲の次（マクロ）ワードのために、ステップ 351 へと戻る。 WX がテスト 324 で少なくとも 1 つの「1」を有する場合は、前述したようにプログラムがステップ 330 へと進む。それ以外では、図 56 の手順が図 51 の手順と同じになる。

【0321】

ループ 590 から 593 の各反復は、テスト 590 でアドレス AD をチェーン終了値（0）と比較して開始する。 $AD > 0$ の場合は、ステップ 591 で層 n のランク値 $NO_n(AD)$ および次アドレス $NX_n(AD)$ をそれぞれ変数 N および M として読み出す。その後、フィルタリングおよび交差ファンクション $FFILT$ をステップ 592 で呼び出してから、ステップ 593 で AD を M で置換する。層 n のレコード・チェーンを完全に検査したら（テスト 590 で $AD = 0$ ）、現（マクロ）ワードの交差リスト WX の計算は終了であり、プログラムは上述したようにテスト 324 へと進む。

【0322】

このファンクション $FFILT$ の流れ図を図 57 に示す。ステップ 600 で実行を開始する時に入力するその引数は、以下の通りである。

10

20

30

40

50

- 図56のステップ592でファンクションFFILTを読み出した場合、コード化層索引 k で、 $k = n$ 。
- 図56のステップ321で取得した通りの k 個のビットマップ・ベクトル $WZ1_q$, $WZ2_q$, $\dots$, WZk_q 。
- 図56のステップ592でファンクションFFILTを読み出した場合、層 k のランク N で $N = NO_n(AD)$ 。
- 層 k データ・コンテナの対応するレコード・アドレス AD 。
- 計算中の交差ビットマップ WX 。

【0323】

テスト601で、ビットマップ・ベクトル WZk_q のランク N のセグメントがゼロのみで構成されているか判断する。イエスの場合は、層 k のランク N に関するコード化データをさらに読み出す必要はなく、したがってステップ602でビットマップ・ベクトル WX を戻して、ファンクションの実行が終了する。 10

【0324】

テスト601でセグメント $WZk_q[N]$ が少なくとも1つの「1」を有する場合は、ステップ603でビットマップ・セグメント $HPk(AD)$ をセグメント変数 H として読み出し、テスト604で交差セグメント $H \text{ AND } WZk_q[N]$ を評価する。この交差セグメントがゼロになっているだけの場合は、同様にコード化データをさらに読み出すのは無駄であり、プログラムは直接ステップ602へと進む。

【0325】

テスト604で $H \text{ AND } WZk_q[N]$ が少なくとも1つの「1」を有していると判明した場合は、下位層のレコード・チェーンへと入る必要がある。ステップ605でその頭部アドレス $F(k-1)(AD)$ を変数 AD' として読み出し、その一方でランク k の剰余 R を0に初期化し、層 $k-1$ のランク N' を $N \times Dk$ に初期化する。ビット $H(R)$ を連続的に検査するループ(テスト606)でビットマップ・セグメント $H = HPk(AD)$ を走査し、ランク $N' = N \times Dk + R$ を考慮すべきか確認する。 $H(R) = 0$ の場合は、ランク N' が現シソーラス項目の層 k のコード化データに入っていないので、これは無視される。すなわち、ステップ607で R を1ユニットだけ増加させ、新しい R がそれでも Dk より小さい場合(テスト608)は、ステップ609で N' も1ユニットだけ増加させてから、テスト606から次の反復へと進む。 20 30

【0326】

テスト606で $H(R) = 1$ である場合は、テスト610でベクトル WZk_q のランク N' のビットを検査し、層 $k-1$ のランク N' が結果リストに入っているか判断する。 $(WZk_q(N') = 0)$ でない場合、プログラムは、ステップ611で AD' を次アドレス $NX(k-1)(AD')$ で置換することにより、層 $k-1$ のレコード・チェーンの次位置にジャンプする。ステップ611の後、プログラムは上述したステップ607へと進む。

【0327】

テスト610で $WZk_q(N') = 1$ である場合、処理は、コード化層 k が1のすぐ上か否か(テスト612)によって決定される。 $k = 2$ の場合は、ビットマップ・セグメント $HP1(AD')$ を読み出して(ステップ613)、ビットごとにブールAND演算でビットマップ・セグメント $WZ1_q[N']$ と組み合わせ、ビットマップ・ベクトル WX のランク N' のセグメント $WX[N']$ を提供する(ステップ614)。テスト612で $k > 2$ である場合は、ステップ615にて引数 k 、 $WZ1_q$, $\dots$, $WZ(k-1)_q$ 、 N' 、 AD' および WX で再帰的ファンクションFFILTを呼び出す。ステップ614または615の後、プログラムは上述したステップ611へと進む。 40

【0328】

テスト608で $R = Dk$ になったら、ビットマップ・セグメント $H = HPk(AD)$ の走査は終了である。次に、ステップ602で更新したビットマップ・ベクトル WX を戻す。

【0329】

(図 5 5 のような) 層 1 のランク・テーブル T N O の使用は、図 5 6 ～図 5 7 の手順と非常に互換性があることが分かる。ステップ 3 5 1 でテーブル T N O のレコードをデフォルト値で初期化し、図 5 7 のステップ 6 1 3 から 6 1 4 の代わりに N ' を T N O (A D ') に書き込み、テスト 5 9 0 で $A D = 0$ の場合は、図 5 5 のループ 3 8 6 から 3 9 1 でテーブル T N O を走査する。

【0 3 3 0】

図 5 1 または図 5 6 の手順は、記憶されているシソーラス・データが、図 5 8 ～図 6 1 で示すように編成したファイルを含む場合は、さらなる最適化を達成することができる。シソーラスごとに、図 5 8 ～図 6 0 に示したタイプのテーブルを記憶し、図 6 1 で示すように、層 n のランク N O n で可能な各値を追加データ・コンテナのレコード・チェーン頭部アドレス F _ A D ' と関連付ける。後者のデータ・コンテナは、図 3 2 または図 3 4 A と同じ層 n のビットマップ・セグメント・データ H P ' 2 = H P 2 を含むが、リンク N X ' 2 は、同じシソーラス項目ではなく同じ層 n ランクに属するレコード・チェーンを定義する。したがって、図 6 1 のデータ・コンテナは、N O 2 列に基づいて図 3 2 または図 3 4 A のそれをソートし、N O 2、N X 2 および F 1 列を削除して、N O 2 に基づいてレコード・チェーンにアドレスを含むよう列 N X ' 2 を追加し、さらにそのレコードが属するシソーラス索引 x が書き込まれた列 P T R を追加することにより取得される。ランク N O 2 ごとに、チェーンの頭部アドレスを F _ A D ' (N O 2) に記憶させる。

【0 3 3 1】

図 5 1 または図 5 6 の手順を開始する前、またはステップ 3 2 1 で反復するごとに、図 6 2 に示した事前フィルタリング処理を適用し、その層 n のランクが、一致するデータ・グラフ識別子リストの層 n のコード化データにないという理由で読み出されないシソーラス項目をマークする。マーキングは、レベル q のシソーラス・ポインタ x_q ごとに 1 つのビット $T_q (x_q)$ を有するマクロワード・レベル q のテーブル T_q によって実行される。これらのビット、さらに層 n のランク $N = N O n$ は、図 6 2 のステップ 6 2 0 でゼロに初期化する。W Z n q のランク N のセグメントがゼロのみで構成されている場合 (テスト 6 2 1) は、テスト 6 2 2 を実行して、可能な最高の層 n のランク $N O n_{max}$ に到達したか判断する。ノーの場合は、ステップ 6 2 3 で N を増加させて、テスト 6 2 1 を繰り返す。テスト 6 2 1 で $W Z n_q [N] \neq 0$ の場合は、ステップ 6 2 4 で頭部アドレス F _ A D ' (N) を変数 A D ' として読み出し、テスト 6 2 5 でチェーン終了値 (0) と比較する。A D ' = 0 の場合は、プログラムがステップ 6 2 2 へと進む。それ以外では、ステップ 6 2 6 でビットマップ・セグメント H P ' n (A D ') および対応する次アドレス値 N X ' n (A D ') をそれぞれ変数 H ' および M ' として読み出す。H ' および W Z n q [N] が共通して「1」を有さない場合 (テスト 6 2 7) は、ステップ 6 2 8 で A D ' を M ' で置換し、テスト 6 2 5 から次の反復を開始する。テスト 6 2 7 で H ' および W Z n q [N] のビットごとのブール A N D 組合せで少なくとも 1 つの「1」がある場合は、図 6 1 の最終列でシソーラス・ポインタ $x_q = P T R (A D ')$ を読み出し、テーブル T_q の対応する位置に「1」を書き込んでから、ステップ 6 2 8 へと進む。

【0 3 3 2】

その後、図 6 3 で示すように、関連するビットマップ・ベクトル W X を計算したら、図 5 1 のループ 3 2 2 から 3 2 7 は、ビット $T_q (x_q)$ をテストする初期フィルタリング・ステップ 6 4 0 によって完成する。このテスト 6 4 0 は、ステップ 3 2 5 で x_q を増加した後にも実行する。テスト 6 4 0 で $T_q (x_q) = 1$ の場合は、前述したようにプログラムがステップ 3 2 2 へと進む。 $T_q (x_q) = 0$ の場合は、直接ステップ 3 2 5 へと進み、これによって空白である交差リスト W X の計算を回避する。

【0 3 3 3】

図 5 6 のステップ 3 5 1 の前に、同じフィルタリング・ステップ 6 4 0 を実行することができる。

【0 3 3 4】

図 5 2 ～図 5 5 に示すファンクション F I N T E R は、図 1 7 のように別個のコード化デ

10

20

30

40

50

ータ・ファイルを各シソーラス・ワードに使用するケースに容易に適用される。図 5 2 のステップ 3 5 2 から 3 5 3 の代わりに、値 $AT\_P(q)(x)$ をワード変数 W に割り当て、ループ索引 i をゼロに初期化する。ステップ 3 5 5 の低密度処理、およびステップ 3 5 7 の層 n の通常密度処理は、図 5 3 および図 5 4 で示した処理と同様である。ループは、共通データ・コンテナ（ループ索引 AD ）では実行されないが、個々のコード化データ・ファイル（図 3 3 のようにループ検索 i ）では実行される。ステップ 3 5 8 の層 $k < n$ の処理は、図 5 5 とは異なり、2 つのループを必要としない。図 6 4 に従ってよい。

【0335】

図 6 4 に示す手順では、ループ索引 i が、シソーラス AT 、マクロワード・レベル q およびワード W に関するコード化データ・ファイルにある層 k のレコードの総数 $imax(AT, q, W, k)$ より低い限り（テスト 3 9 4）、ステップ 3 9 5 から 3 9 9 が実行される。ステップ 3 9 5 では、ランク $AT\_P(q)\_W\_NOk(i)$ を整数変数 N に関連付ける。その後のステップ 3 9 6 では、ビットマップ・ベクトル WZ のランク N のセグメント $WZ[N]$ をテストする。 $WZ[N]$ が少なくとも 1 つの「1」を有する（ $WZ[N] \neq 0$ ）場合は、ビットマップ・セグメント $AT\_P(q)\_W\_HPk(i)$ を読み出し（ステップ 3 9 7）、ビットごとのブール AND 演算でビットマップ・セグメント $WZ[N]$ と組み合わせて、ビットマップ・ベクトル WX のランク N のセグメント $WX[N]$ を提供する（ステップ 3 9 8）。ステップ 3 9 8 の後、またはテスト 3 9 6 で $WZ[N] = 0$ の場合に実行されるステップ 3 9 9 では、ループ索引 i を 1 ユニットだけ増加させてから、テスト 3 9 4 へと戻る。関連するコード化データを完全に検査したら、すなわち、テスト 3 9 4 で $i = imax(AT, q, W, k)$ の場合は、ループが終了である。

【0336】

上述した手順は、SQL 問合せの特徴に基づき、図 5 1 または図 5 6 のステップ 3 3 5 で異なるタイプの措置を含むことができる。

【0337】

比較的簡単なタイプの SQL 問合せでは、1 つの属性の値のリスト（例えば、特定の基準を満たす全顧客の名前）が必要である。このようなケースでは、図 5 1 または図 5 6 の走査が、その属性に関するシソーラスでのみ、 $QA = 0$ で実行され、ステップ 3 3 5 の措置は、単に、個々のワード・シソーラスの位置 x_0 にあるワード $AT(x_0)$ を読み出して（実際、コード化データが図 1 7 に示すように記憶されていると、ワード $AT(x_0)$ は直前に読み出されている）、このワード $AT(x_0)$ を出力テーブルに書き込むか、それを印刷するだけである。このようにして生成されたワード・リストは、昇順で自動的にソートされることが分かる。反対の順序が必要な場合は、シソーラスを反対方向に走査することができる。

【0338】

SQL が、属性 AT に関する $SELECT$ 文節に $DISTINCT$ キーワードを有する場合は、ステップ 3 3 5 でワード $AT(x_0)$ を 1 回出力する。そうでない場合、または $SELECT$ 文節がキーワード ALL を有する場合は、ステップ 3 3 5 で WX の非ゼロ・ビットごとにワード $AT(x_0)$ を 1 回出力することができる。この非ゼロ・ビットをカウントして、マッチングデータ・グラフ中のワード $AT(x_0)$ の発生数を提供することもできる。

【0339】

正確さを下げて属性の値が必要である場合は、 $QA > 0$ の条件で図 5 1 または図 5 6 に示したようにシソーラスを走査し、それによって $q < QA$ のレベル q シソーラスから関係ない詳細を取得するメモリ・アクセスを回避することができる。例えば、日付の属性を年単位で表現する必要がある場合は、4 の切り捨て長に対応するレベル QA で、図 5 1 または図 5 6 の走査を停止することができる。

【0340】

SQL 問合せは、 $SELECT$ および $FROM$ 文節にある幾つかの属性を必要とすることが多い。同じデータ・グラフに属する属性値間の接続を維持するため、何らかの形式の索

引付けが必要である。可能な1つの方法は、RAM103内に、仮想フラットファイルと尾内だけの行を有する出力テーブル、および結果リストの属性値を受ける個々の列を含むための作業ゾーンを確保することである。出力テーブルのメモリ・ロケーションを、デフォルト値で初期化する。上述した属性値 $AT(x_0)$ または $QA > 0$ の場合はその接頭語 $AT(x_{QA})$ を、図51または図56に示すステップ335の出現の出力テーブルに書き込む。ステップ335のこのような書き込み作業は、ビットマップ・ベクトル WX の非ゼロ・ビットで示された出力テーブルの任意の行で実行される。出力データは、最終的に、出力テーブルから空白行（まだデフォルト値を含む行）を削除することによって生成される。

【0341】

10

図65は、このようなケースでステップ335を発展させ、適宜、ワード $W = AT(x_{QA})$ を出力テーブルの列 OT_AT に書き込む方法を示す。ステップ400で行ポインタ j をゼロに初期化し、ワード W を（以前に実行されていなかったら）ロードする。ビット $WX(j)$ が1であるたびに（テスト401）、ワード W を出力テーブルの行 j および列 AT に書き込む（ステップ402）。テスト403で行ポインタ j を最大値 j_{max} と比較し、 $j < j_{max}$ である場合は増加させる（ステップ404）。テスト403で $j = j_{max}$ になると、プログラムはステップ335の措置を終了させたことになる。

【0342】

例4：

例3の問合せ基準を考察し、表示するよう要求された属性が事故日、顧客名、および保険日であると仮定する。図37～図38に関して検討する例3では、マッチングデータ・グラフのビットマップ（図41のステップ246の出力）は、図8でチェックするように、 $Res = 101100001000$ である。この例では、図66が、前述したような出力テーブルの内容を示す。

20

【0343】

上述した出力テーブルは、RAM103に都合よく確保するには大きすぎることもある。実際のデータベースでは、仮想フラットファイルの行数が比較的大きく（例えば、数百万）、（含むべき属性が多すぎるか、一部が比較的多数の文字を使用しているために）出力テーブルの1行にある文字が多すぎる場合は、出力テーブルが法外に大きくなることもある。この潜在的問題に対処するため、幾つかの解決法がある。

30

【0344】

その1つは、図65のステップ402で、シソーラス行 ID 、 x_{QA} （整数）を（マクロ）ワード $AT(x_{QA})$ ではなく出力テーブルに書き込むことである。関連する全てのシソーラスを走査したら、空白でない出力テーブルの行を選択し、シソーラス行 ID から属性値を検索する。これにより、出力テーブルの列の幅が狭くなる。ワード $AT(x_{QA})$ が往々にして、はるかに多くの文字を必要とするからである。

【0345】

図67は、例4のケースで、このような出力テーブルの内容を示し、シソーラスは図10A～図10Gと同様にソートされる。

【0346】

40

代替的または累積的な別の解決法は、RAM103の索引を使用して、整数アドレスを各データ・グラフまたはフラットファイル行 ID に関連付けることである。最初に、デフォルト・アドレスを全データ・グラフに関連付ける。その1つが、ステップ335で WX の対応するビットで初めて「1」で指定されたら（すなわち、第1のシソーラスを走査したら）、カウンタを増加させて取得した新しいアドレスに割り当てられる。このアドレスは、その後のシソーラスの走査時にこのデータ・グラフが再度指定されると、索引から検索される。この整数アドレスは、RAM103に記憶された出力テーブルの行 ID であり、属性値または接頭語 $AT(x_{QA})$ 、またはそのシソーラス行 ID 、 x_{QA} が書き込まれた行の数が減少する。非空白行は連続し、したがって行の総数を大幅に減少させることができる。この圧縮した出力テーブルは、最終的に読み出されて、結果を表示する。

50

【0347】

図68は、このような索引および出力テーブルの内容を示し、例4のケースではシソーラス行IDを含む。

【0348】

図69および図70は、それぞれ第1のシソーラスおよびその後のシソーラスの走査時にステップ335を発展させる方法を示す。同じ参照番号で示したステップ400、401、403、404は、図65のそれと同じである。図69では、ビットWX(j)が1であれば、カウンタ値m(図51または図56のステップ320で0に初期化される)を行jの索引IND(j)に割り当て(ステップ410)、シソーラス・ポインタ x_{QA} (またはワード $W = AT(x_{QA})$)を出力テーブルの行jおよび列ATに書き込み(ステップ411)、カウンタ値mを増加させる(ステップ412)。第1のシソーラスの走査が終了したら、mはマッチングデータ・グラフの数を表す。図70では、ビットWX(j)が1の場合は、行jの索引IND(j)をポインタm'として検索し(ステップ413)、シソーラス・ポインタ x_{QA} (またはワードW)を出力テーブルの行m'および列ATに書き込む(ステップ414)。

10

【0349】

出力テーブルは、SQL問合せがGROUP BY、ORDER BYまたは同様の文節を有する場合、その列の内容に基づいて容易にソートされる。このようなソート作業は、複数の属性に関して階層的に実行することができる。出力テーブルを構築中に第1のソート基準が自動的に見たされるよう、階層中の最も重要な属性には、図51または図56に示すような第1のシソーラス走査を実行することが好ましい。以前の属性と共通の値を有する出力テーブルの各部分で、残りの属性に基づくソートを実行する。

20

【0350】

ソートは、出力テーブルの列が図68のようにシソーラス行ID、 x_{QA} を含む場合、整数リストのソートしか含まないので、特に簡単である。

【0351】

特定の属性、特に数値フィールドでは、(リンク・テーブルがあれば)明示的属性値をリンク・テーブルに記憶できることが、以前に示されている。図66、図67または図68に示すタイプの出力テーブルは、このような属性の列を有する必要がある。属性を表示するか、他の方法で調べる必要がある場合、その変数は、出力テーブルの非空白行(図66~図67)または出力テーブル索引の有効ポインタ(図68)に対応する行にある(すなわち、これと同じ行IDを有する)リンク・テーブルから検索することができる。

30

【0352】

SQL問合せでは、特にデータ・ウェアハウス用途で、マッチングデータ・レコードの属性値に関する計算も必要である。このような計算は、図66、図67または図68に示すタイプの出力テーブルのデータから実行することができる。

【0353】

例5:

例4から、事故日と保険日間の時間差の(算術)平均値は、日数として要求し、表現すると仮定する。出力テーブルの非空白行ごとに、プログラムは第1列と第3列間の差を日数で計算する。この差を累積し、その結果を非空白行の数(4)で割って、所望の平均値を取得する。

40

【0354】

実際、この平均値は、1つだけのメモリ・ロケーションに減少させた出力テーブルで計算することができる。すなわち、事故日シソーラスを走査する場合、図51または図56のステップ335で、属性基準日からの日数として表現される属性値にWXの非ゼロ・ビット数を掛け、減少させた出力テーブルのメモリ・ロケーションに記憶されている累積変数V(ステップ320で、0に初期化)に加算する。次に、保険日シソーラスを走査する場合、ステップ335で、同じ基準日からの日数として表現される属性値にWXの非ゼロ・ビット数を掛け、ステップ335でVから引く。最後に、結果のVを結果ビットマップR

50

e s の非ゼロ・ビット数で割って、所望の平均値を提供する。

【0355】

しかし、RAM103では、その種の計算のため、特に所望の量が属性値に対して直線でない場合（例えば、例5で算術平均ではなく二次または相乗平均値を要求する場合）、複数のメモリ・ロケーションを有する出力または計算テーブルが往々にして有用である。

【0356】

計算テーブルは出力テーブルの特殊なケースであり、上述した出力テーブルと同様の構造を有する。これは、（図66～図67の出力テーブルのように）仮想フラットファイルと同じ数の行を有することができる。あるいは、図68のそれと同じ索引に関連付けてもよい。1つのメモリ・ロケーションを有する出力テーブルの上記例と同様、行を1つだけ有してもよい。計算テーブルの各列は、実行すべき計算に使用するオペランドの値を含むためのものである。計算の複雑さに応じて、1つまたは複数の列が必要であるが、たいていの場合、1つの列で十分である。

10

【0357】

計算に参与する値を有する属性は、図51または図56に関して説明するように、そのシソーラスが連続的に走査される。ステップ335は、1つの列、計算テーブルCTが、および仮想フラットファイルと同じ数の行を有する場合、図71に示すように展開することができる（索引がある場合は、図69～図70のように取り扱うことができる）。図71では、ステップ400、401、403および404が、図65のそれと同じである。ステップ401でビットWX(j)が1の場合は、ステップ416で、行jにある計算テーブルの内容CT(j)をオペランドYに割り当て、次にオペランドYおよび現（マクロ）ワードW=AT(x_{QA})のファンクションfを計算して、ステップ417で新しい内容CT(j)として保存する。

20

【0358】

数学的ファンクションfは、実行すべき計算、および走査中のシソーラスに基づいて計算する。再び例5に言及すると、事故日を最初に走査する場合、ファンクションf(Y, W)は、フォーマットyyyy mm ddで表現された日付Wを基準日からの日数に変換することでよい（したがってWのみのファンクションである）。保険日シソーラスを走査する場合、ファンクションf(Y, W)は、同じ変換を日付Wに当てはめ、Yからの結果を引く。その後、CTの非空白行の平均値（算術、二次、相乗）を計算して、所望の出力結果を提供する。計算テーブルの列からは、例えば、統計、会計、または実際の計算など、他の種類の包括的計算を実行することができる。

30

【0359】

所望の正確さが、関係する属性のうち少なくとも1つの個別ワードによって可能な正確さより低い場合は、このタイプの計算にマクロワードを使用すると有利である。

【0360】

仮想フラットファイルの区分

大型システムの場合は、仮想フラットファイルを、それぞれが所定数の行で構成された幾つかの部分またはブロックに区分すると、往々にして有利である。データ・グラフを、その識別子（フラットファイル行ID）に基づいてブロックに分散する。

40

【0361】

各シソーラスは、対応するシソーラス・セクションに分割することが好ましく、これによって各セクションは、対応する仮想フラットファイル・ブロックに含まれるフラットファイル行IDリストを備えた項目を有する。属性に割り当てられた1つのワードを伴う完全なフラットファイル行IDリストは、そのワードの対応するシソーラス・セクションの項目で表されるリストの集合体である。したがって、シソーラス項目の完全なフラットファイル行IDリストは、仮想フラットファイルと同様に区分され、シソーラス・セクションに対応するサブリストに分割される。

【0362】

属性のシソーラス索引ファイルは、全セクションに共通でよい。セクションごとに別個の

50

索引ファイルを設けてもよい。

【0363】

ブロックごとに、図38～図71に関して上述したように、SQL問合せの処理のステップ191から193（図36）を実行する。このようにして取得した結果をマージして、応答を表示する。

【0364】

異なるブロックに関する問合せの処理は、順次、または並行して実行することができる。

【0365】

順次処理では、最適の処理速度を得るためのRAMの可用性を効果的に制御することができる。RAM回路の費用は、現在では重要だと考えられていないが、任意のマシンは特定量のRAMの容量が使用可能であり、これが上述した出力または計算テーブル用にRAMスペースを確保するための制約となる。制約を受ける可能性が高い場合は、仮想フラットファイルと区分すると、そのテーブルのサイズ（図65および図69～図71のjmax）がそのまま減少する。 10

【0366】

したがって、本発明を実行するための特定のマシンは、jmax、すなわち、ブロック・サイズの選択に従う。仮想フラットファイル・ブロックは、選択したサイズのパラメータに基づいてサイズ決定され、対応するシソーラス・セクションは、図19のステップ122から126に関して示したように、セクションごとに次々と構築する。

【0367】

このように問合せ処理エンジンのサイズを決定すると、全ての段階で最適なアルゴリズムを使用できながら、RAM103とハード・ドライブ105の間で中間データを交換する必要がない。 20

【0368】

並行処理を使用すると、さらなる加速が達成される。問合せ処理を、仮想フラットファイル・ブロックごとに幾つかのプロセッサに分散させる。

【0369】

全てのブロックが同じサイズjmaxを有するという特殊なケースで、並行問合せ処理エンジンで可能なアーキテクチャを図72に示す。マッチングユニット700の数Mを、通信ネットワーク702を通して問合せサーバ701に接続する。各マシン・ユニット700は、図18に示すタイプのプロセッサ・システムであってもよい。これは、ブロックに関連付けたシソーラス・セクションを記憶するため、ハード・ドライブなどの記憶装置703を有する。図9に示すタイプのリンク・テーブルを使用する場合は、仮想フラットファイルと同じ方法でこれをブロックに区分し、各ブロックを対応するマッチングユニットに記憶する。サーバ701がマンマシン・インタフェースを提供する。これはSQLのWHERE文節の問合せ基準を図37に示すタイプの本に変換し、これが所望の出力の記述とともにM個のマッチングユニット700に提供される。各ユニット700は、図36のステップ191から193に従って自身の部分の仕事を実行し、その応答をサーバ701へと戻す。後者は、様々なマッチングユニットからの結果をコンパイルし、全体的な応答をユーザに提供する。ステップ191の分析を実行するため、各マッチングユニット700はそのシソーラス・セクションを使用する。 30 40

【0370】

あるいは、問合せ基準の分析は、サーバ701が大域シソーラスによって中央で実行することができ、各大域シソーラスは、全（マクロ）ワードに共通であり、M個の記憶ユニット703の識別子サブリストへのポインタを含むため、M個の列を有する。分析段階の最後に、ステップ192から193を実行するため、関連するポインタをマッチングユニット700にアドレス指定する。

【0371】

更新サーバ704は、問合せサーバ701と同じマシンであってもよいが、これもネットワーク702に接続されて、様々なブロックに関するVDGを生成し、維持する。これは 50

、RDBMSのデータ・テーブルの変化をモニタし、シソーラス・セクションに必要な変更を実行するため、シソーラス更新命令をユニット700に経路指示する。

【0372】

上述した並行システムは、データ・グラフの数が現在の最大値（図72の図では $M \times j_{max}$ ）に近づく場合に、容易に実行される。それには、新しい仮想フラットファイル・ブロックを扱うために、さらなるマッチングユニットを追加する必要がある、そのサイズは、以前のブロックと同じでも異なってもよく、さらにサーバ701、704の経路指示および結果コンパイル容量を再編成する必要がある。再編成は、以前から存在したマッチングユニットには完全にトランスペアレントである。したがって、最小の費用でシステム容量を増加させることができる。システムを停止する必要もない。

10

【図面の簡単な説明】

【0373】

【図1】従来のリレーショナル・データベース・システムで通常使用するようなデータ構造の例を示す図である。

【図2】従来のリレーショナル・データベース・システムで通常使用するようなデータ構造の例を示す図である。

【図3】従来のリレーショナル・データベース・システムで通常使用するようなデータ構造の例を示す図である。

【図4】図1～図3の例のデータ・テーブル木を表す図である。

【図5】図4の木および図1のデータで構築したデータ・グラフを示す図である。

20

【図6】図4の木および図2のデータで構築したデータ・グラフを示す図である。

【図7】図4の木および図3のデータで構築したデータ・グラフを示す図である。

【図8】図1～図3のデータ・テーブルのフラットファイル表示の図である。

【図9】本発明の実施形態で使用するような連結テーブルを示す図である。

【図10A】図1～図3のデータ・テーブルに対応するシソーラスの内容を示す図である。

【図10B】図1～図3のデータ・テーブルに対応するシソーラスの内容を示す図である。

【図10C】図1～図3のデータ・テーブルに対応するシソーラスの内容を示す図である。

30

【図10D】図1～図3のデータ・テーブルに対応するシソーラスの内容を示す図である。

【図10E】図1～図3のデータ・テーブルに対応するシソーラスの内容を示す図である。

【図10F】図1～図3のデータ・テーブルに対応するシソーラスの内容を示す図である。

【図10G】図1～図3のデータ・テーブルに対応するシソーラスの内容を示す図である。

【図10H】図1～図3のデータ・テーブルに対応するシソーラスの内容を示す図である。

40

【図11A】図10Aのシソーラスの他の表示を示す図である。

【図11G】図10Gのシソーラスの他の表示を示す図である。

【図11H】図10Hのシソーラスの他の表示を示す図である。

【図12A】図10Aのシソーラスの他の表示を示す図である。

【図12G】図10Gのシソーラスの他の表示を示す図である。

【図12H】図10Hのシソーラスの他の表示を示す図である。

【図13A】図10Aのシソーラスの他の表示を示す図である。

【図13G】図10Gのシソーラスの他の表示を示す図である。

【図13H】図10Hのシソーラスの他の表示を示す図である。

【図14A】図10Aのシソーラスの他の表示を示す図である。

50

- 【図 1 4 G】図 1 0 G のシソーラスの他の表示を示す図である。
- 【図 1 4 H】図 1 0 H のシソーラスの他の表示を示す図である。
- 【図 1 5】図 1 4 A、図 1 4 G および図 1 4 H のシソーラスに関連するデータ・コンテナに記憶されるデータを示す図である。
- 【図 1 6】図 1 4 A、図 1 4 G および図 1 4 H のシソーラスに関連するデータ・コンテナに記憶されるデータを示す図である。
- 【図 1 7】図 1 0 A ～図 1 4 A のシソーラスの可能な別の構造を示す図である。
- 【図 1 8】本発明を実施するのに適切なコンピュータ・システムのブロック図である。
- 【図 1 9】本発明の実施形態によるデータ・グラフ作成手順を示す流れ図である。
- 【図 2 0】図 1 9 の段階 1 2 4 に適用することができる手順を示す流れ図である。 10
- 【図 2 1】図 2 0 のステップ 1 3 6 に適用することができる手順を示す流れ図である。
- 【図 2 2】図 2 0 のステップ 1 3 6 に適用することができる手順を示す流れ図である。
- 【図 2 3】2 つの連続するコード化層で図 2 0 のステップ 1 3 6 に適用することができる別の手順を示す流れ図である。
- 【図 2 4】2 つの連続するコード化層で図 2 0 のステップ 1 3 6 に適用することができる別の手順を示す流れ図である。
- 【図 2 5】図 1 ～図 3 の例から構築したシソーラスを記憶する方法を示すテーブルである。
- 【図 2 6】図 1 ～図 3 の例から構築したシソーラスを記憶する方法を示すテーブルである。 20
- 【図 2 7】図 1 ～図 3 の例から構築したシソーラスを記憶する方法を示すテーブルである。
- 【図 2 8】図 1 ～図 3 の例から構築したシソーラスを記憶する方法を示すテーブルである。
- 【図 2 9】図 1 ～図 3 の例から構築したシソーラスを記憶する方法を示すテーブルである。
- 【図 3 0】図 1 ～図 3 の例から構築したシソーラスを記憶する方法を示すテーブルである。
- 【図 3 1】図 1 ～図 3 の例から構築したシソーラスを記憶する方法を示すテーブルである。 30
- 【図 3 2】図 1 ～図 3 の例から構築したシソーラスを記憶する方法を示すテーブルである。
- 【図 3 3】シソーラスを図 1 7 に示すように記憶した場合に、図 2 0 のステップ 1 3 5 および 1 3 6 を実行する代替方法を示す流れ図である。
- 【図 3 4 A】図 3 1 ～図 3 2 のテーブルの代替実施形態を示すテーブルである。
- 【図 3 4 B】図 3 1 ～図 3 2 のテーブルの代替実施形態を示すテーブルである。
- 【図 3 4 C】図 3 4 A および図 3 4 B のテーブルの別の表示である。
- 【図 3 4 D】図 3 4 A および図 3 4 B のテーブルの別の表示である。
- 【図 3 5】図 3 4 A および図 3 4 B で示したタイプのテーブルの管理に適用することができる手順を示す流れ図である。 40
- 【図 3 6】本発明の実施形態による問合せ処理手順の一般的流れ図である。
- 【図 3 7】図 1 ～図 3 の例に関する問合せ木の例を示す図である。
- 【図 3 8】図 3 7 の問合せ木の分析によって獲得された拡張問合せ木を示す別の図である。
- 【図 3 9】問合せ木を分析する手順を示す流れ図である。
- 【図 4 0 A】図 3 9 の手順に関する再帰関数の流れ図である。
- 【図 4 0 B】図 3 9 の手順に関する再帰関数の流れ図である。
- 【図 4 1】図 3 8 に図示されたような拡張問合せ木に基づき、マッチングデータ・グラフを識別する流れ図手順である。
- 【図 4 2】図 4 1 の手順で呼ばれる再帰関数 F N O D E の流れ図である。 50

- 【図 4 3】図 4 2 のステップ 2 6 2 で実行される手順を示す流れ図である。
- 【図 4 4】図 4 2 のステップ 2 6 4 で実行される手順を示す流れ図である。
- 【図 4 5】図 4 2 のステップ 2 6 5 で実行される手順を示す流れ図である。
- 【図 4 6】図 4 2 のステップ 2 6 5 の手順の代替実施形態を示す流れ図である。
- 【図 4 7】シソーラスを図 3 4 A および図 3 4 B で示すように記憶した場合、図 4 2 のステップ 2 6 5 の手順の別の代替実施形態を示す流れ図である。
- 【図 4 8】図 4 7 の手順で呼ばれる再帰関数 F I L T の流れ図である。
- 【図 4 9】シソーラスを図 1 7 で示すように記憶した場合、図 4 2 のステップ 2 6 5 の手順の別の代替実施形態を示す流れ図である。
- 【図 5 0】図 4 2 の関数に使用する葉処理の変形の流れ図である。 10
- 【図 5 1】データベース問合せに関連する属性値を検索するために、任意の属性に関してシソーラスを走査するのに適用することができる手順を示す流れ図である。
- 【図 5 2】図 5 1 の手順で言及される関数 F I N T E R の流れ図である。
- 【図 5 3】図 5 2 のステップ 3 5 5 で実行する手順を示す流れ図である。
- 【図 5 4】図 5 2 のステップ 3 5 7 で実行する手順を示す流れ図である。
- 【図 5 5】図 5 2 のステップ 3 5 8 で実行する手順を示す流れ図である。
- 【図 5 6】シソーラスを図 3 4 A および図 3 4 B で示すように記憶した場合、データベース問合せに関連する属性値を検索するために、任意の属性に関してシソーラスを走査するのに適用することができる代替手順を示す流れ図である。
- 【図 5 7】図 5 6 の手順で呼ばれる再帰関数 F F I L T の流れ図である。 20
- 【図 5 8】図 2 5 ～図 3 4 のテーブルと協働で記憶することができるテーブルを示す。
- 【図 5 9】図 2 5 ～図 3 4 のテーブルと協働で記憶することができるテーブルを示す。
- 【図 6 0】図 2 5 ～図 3 4 のテーブルと協働で記憶することができるテーブルを示す。
- 【図 6 1】図 2 5 ～図 3 4 のテーブルと協働で記憶することができるテーブルを示す。
- 【図 6 2】図 5 1 と同様のシソーラス走査に先だって使用することができる事前フィルタリング手順を示す流れ図である。
- 【図 6 3】図 6 2 による事前フィルタリングを考慮に入れるために適用される、図 5 1 によるシソーラス走査手順の一部を示す流れ図である。
- 【図 6 4】シソーラスを図 1 7 で示すように記憶した場合、図 5 2 のステップ 3 5 8 に適用することができる代替手順を示す流れ図である。 30
- 【図 6 5】図 5 1 のステップ 3 3 5 に適用することができる手順を示す流れ図である。
- 【図 6 6】問合せ応答を提供するために使用する例示としての出力テーブルの内容を示す図である。
- 【図 6 7】問合せ応答を提供するために使用する例示としての出力テーブルの内容を示す図である。
- 【図 6 8】出力テーブルで可能な別の構造を示す図である。
- 【図 6 9】図 6 8 で示すタイプの出力テーブルを構築するために、図 5 1 のステップ 3 3 5 に適用することができる手順を示す流れ図である。
- 【図 7 0】図 6 8 で示すタイプの出力テーブルを構築するために、図 5 1 のステップ 3 3 5 に適用することができる手順を示す流れ図である。 40
- 【図 7 1】計算テーブルによってデータベース・システムで計算を実行するために、図 5 1 のステップ 3 3 5 に適用することができる手順を示す流れ図である。
- 【図 7 2】本発明を実施するために適切な別のコンピュータ・システムのブロック図である。
- 【符号の説明】
- 【0 3 7 4】
- 1 0 0 バス
- 1 0 1 C P U
- 1 0 2 R O M
- 1 0 3 R A M

- 1 0 4 マンマシン・インタフェース
- 1 0 5 ハード・ドライブ
- 1 0 6 磁気ディスク

【図 1】

顧客 行ID	名前	生年	性別
0	Oscar	1959	M
1	André	1976	M
2	Ariane	1965	F
3	Laure	1976	F
4	Max	1947	M

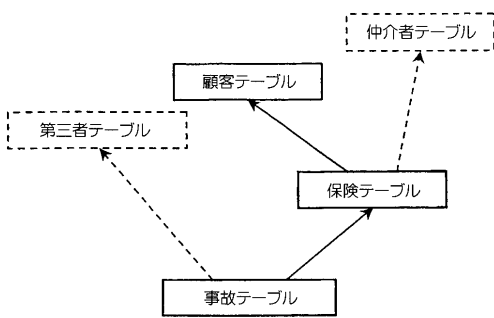
【図 2】

保険 行ID	タイプ	日付	顧客リンク
0	自動車	1998 06 24	2
1	自動車	1998 09 09	1
2	住宅	1998 12 11	4
3	自動車	1999 01 10	4
4	住宅	1999 01 31	2
5	自動車	1999 02 01	3
6	住宅	1999 02 01	3

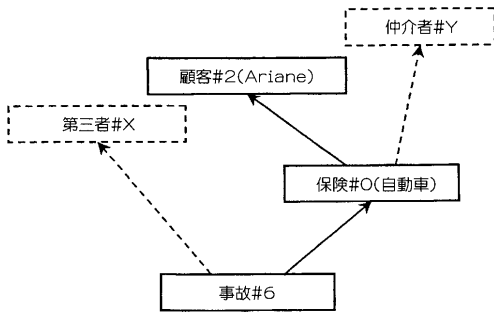
【図 3】

事故 行ID	日付	金額	保険リンク
0	1998 10 03	1 000	1
1	1999 04 18	10 000	4
2	1999 04 18	800	3
3	1999 06 12	1 300	1
4	1999 08 31	300	3
5	1999 12 08	2 200	5
6	1999 12 09	1 000	0
7	2000 02 25	6 000	4
8	2000 03 20	800	1

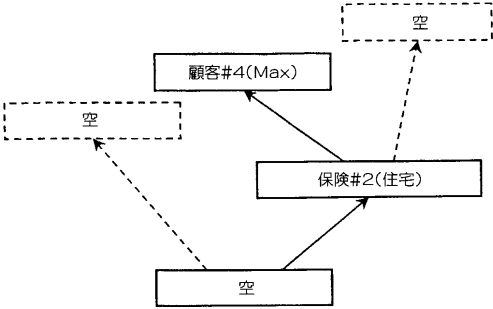
【図 4】



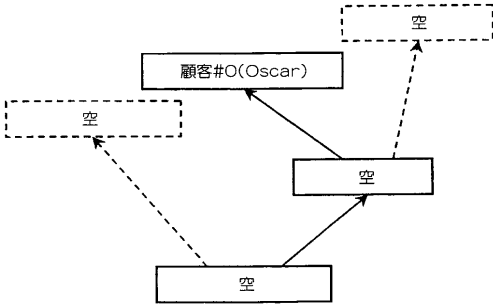
【図 5】



【図 6】



【図 7】



【図 9】

リンク・テーブル フラットファイル 行ID	顧客リンク			保険リンク			事故リンク		
	0	1	2	0	1	2	0	1	2
0	1	2	4	1	4	3	0	1	2
1	2	4	3	1	3	1	1	2	4
2	4	1	1	3	5	0	2	4	3
3	1	4	3	0	4	1	3	1	2
4	3	2	2	1	2	6	1	4	3
5	2	1	4	2	3	1	2	4	0
6	4	3	1	4	1	2	3	1	2
7	1	2	3	1	4	3	1	2	4
8	2	4	1	2	3	1	2	4	0
9	4	1	2	3	1	2	3	1	2
10	3	2	4	1	4	3	1	2	4
11	0	1	2	2	3	1	2	4	0

【図 8】

フラットファイル 行ID	顧客名		生年		性別		保険タイプ		保険日		事故日		金額	
	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	André	1976	M	自動車	1998 09 09	1998 10 03	1 000							
1	Ariane	1965	F	住宅	1999 01 31	1999 04 18	10 000							
2	Max	1947	M	自動車	1999 01 10	1999 04 18	800							
3	André	1976	M	自動車	1998 09 09	1999 06 12	1 300							
4	Max	1947	M	自動車	1999 01 10	1999 08 31	300							
5	Laure	1976	F	自動車	1999 02 01	1999 12 08	2 200							
6	Ariane	1965	F	自動車	1998 06 24	1999 12 09	1 000							
7	Ariane	1965	F	住宅	1999 01 31	2000 02 25	6 000							
8	André	1976	M	自動車	1998 09 09	2000 03 20	800							
9	Max	1947	M	住宅	1998 12 11		空							
10	Laure	1976	F	住宅	1999 02 01		空							
11	Oscar	1959	M	空			空							

【図 10 A】

顧客名シソーラス		
名前		フラットファイル 行IDリスト
0	空	-
1	André	{0,3,8}
2	Ariane	{1,6,7}
3	Laure	{5,10}
4	Max	{2,4,9}
5	Oscar	{11}

【図 10 B】

顧客生年シソーラス		
生年		フラットファイル 行IDリスト
0	空	-
1	1947	{2,4,9}
2	1959	{11}
3	1965	{1,6,7}
4	1976	{0,3,5,8,10}

【図 10 C】

顧客性別シソーラス		
性別		フラットファイル 行IDリスト
0	空	-
1	F	{1,5,6,7,10}
2	M	{0,2,3,4,8,9,11}

【図 10D】

保険タイプ・シソーラス

	タイプ	フラットファイル 行IDリスト	ビットマップ
0	空	{11}	000000000001
1	自動車	{0,2,3,4,5,6,8}	101111101000
2	住宅	{1,7,9,10}	010000010110

【図 10E】

保険日シソーラス

	日付	フラットファイル 行IDリスト	ビットマップ
0	空	{11}	000000000001
1	1998 06 24	{6}	000000100000
2	1998 09 09	{0,3,8}	100100001000
3	1998 12 11	{9}	000000000100
4	1999 01 10	{2,4}	001010000000
5	1999 01 31	{1,7}	010000010000
6	1999 02 01	{5,10}	000001000010

【図 10F】

事故日シソーラス

	日付	フラットファイル 行IDリスト	ビットマップ
0	空	{9,10,11}	000000000111
1	1998 10 03	{0}	100000000000
2	1999 04 18	{1,2}	010000000000
3	1999 06 12	{3}	000100000000
4	1999 08 31	{4}	000010000000
5	1999 12 08	{5}	000001000000
6	1999 12 09	{6}	000000100000
7	2000 02 25	{7}	000000010000
8	2000 03 20	{8}	000000001000

【図 11G】

事故金額シソーラス

金額	ビットマップ	L1ランク	L1セグメント
空	000 000 000 111	{3}	{111}
300	000 010 000 000	{1}	{010}
800	001 000 001 000	{0,2}	{001,001}
1 000	100 000 100 000	{0,2}	{100,100}
1 300	000 100 000 000	{1}	{100}
2 200	000 001 000 000	{1}	{001}
6 000	000 000 010 000	{2}	{010}
10 000	010 000 000 000	{0}	{010}

【図 11H】

事故金額マクロワード(／3)シソーラス

接頭語	ビットマップ	L1ランク	L1セグメント
0	001 010 001 000	{0,1,2}	{001,010,001}
1	100 100 100 000	{0,1,2}	{100,100,100}
2	000 001 000 000	{1}	{001}
6	000 000 010 000	{2}	{010}
10	010 000 000 000	{0}	{010}

【図 12A】

顧客名シソーラス

名前	L1ランク	L1ビットマップ	L2ランク	L2セグメント
空	-	00 00	-	-
André	{0,1,2}	11 10	{0,1}	{11,10}
Ariane	{0,2}	10 10	{0,1}	{10,10}
Laure	{1,3}	01 01	{0,1}	{01,01}
Max	{0,2,3}	10 11	{0,1}	{10,11}
Oscar	{3}	00 01	{1}	{01}

【図 10G】

事故金額シソーラス

	金額	フラットファイル 行IDリスト	ビットマップ
0	空	{9,10,11}	000000000111
1	300	{4}	000010000000
2	800	{2,8}	001000001000
3	1 000	{0,6}	100000100000
4	1 300	{3}	000100000000
5	2 200	{5}	000001000000
6	6 000	{7}	000000010000
7	10 000	{1}	010000000000

【図 10H】

事故金額(切り捨て長3)シソーラス

金額接頭語	フラットファイル 行IDリスト	ビットマップ
0	{2,4,8}	001010001000
1	{0,3,6}	100100100000
2	{5}	000001000000
6	{7}	000000010000
10	{1}	010000000000

【図 11A】

顧客名シソーラス

名前	ビットマップ	L1ランク	L1セグメント
空	000 000 000 000	-	-
André	100 100 001 000	{0,1,2}	{100,100,001}
Ariane	010 000 110 000	{0,2}	{010,110}
Laure	000 001 000 010	{1,3}	{001,010}
Max	001 010 000 100	{0,2,3}	{001,010,100}
Oscar	000 000 000 001	{3}	{001}

【図 12G】

事故金額シソーラス

金額	L1ランク	L1ビットマップ	L2ランク	L2セグメント
空	{3}	00 01	{1}	{01}
300	{1}	01 00	{0}	{01}
800	{0,2}	10 10	{0,1}	{10,10}
1 000	{0,2}	10 10	{0,1}	{10,10}
1 300	{1}	01 00	{0}	{01}
2 200	{1}	01 00	{0}	{01}
6 000	{2}	00 10	{1}	{10}
10 000	{0}	10 00	{0}	{10}

【図 12H】

事故金額マクロワード(／3)シソーラス

接頭語	L1ランク	L1ビットマップ	L2ランク	L2セグメント
0	{0,1,2}	11 10	{0,1}	{11,10}
1	{0,1,2}	11 10	{0,1}	{11,10}
2	{1}	01 00	{0}	{01}
6	{2}	00 10	{1}	{10}
10	{0}	10 00	{0}	{10}

【図 13A】

顧客名シソーラス

名前	L1ランク	L1セグメント	L2ランク	L2セグメント
空	-	-	-	-
André	{0,1,2}	{100,100,001}	{0,1}	{11,10}
Ariane	{0,2}	{010,110}	{0,1}	{10,10}
Laure	{1,3}	{001,010}	{0,1}	{01,01}
Max	{0,2,3}	{001,010,100}	{0,1}	{10,11}
Oscar	{3}	{001}	{1}	{01}

【図 1 3 G】

事故金額シソーラス

金額	L1ランク	L1セグメント	L2ランク	L2セグメント
空	{3}	{111}	{1}	{01}
300	{1}	{010}	{0}	{01}
800	{0,2}	{001,001}	{0,1}	{10,10}
1 000	{0,2}	{100,100}	{0,1}	{10,10}
1 300	{1}	{100}	{0}	{01}
2 200	{1}	{001}	{0}	{01}
6 000	{2}	{010}	{1}	{10}
10 000	{0}	{010}	{0}	{10}

【図 1 4 G】

事故金額シソーラス

金額	L1冒頭	L1最終	L2冒頭
空	13	13	21
300	4	4	22
800	9	10	16
1 000	3	19	18
1 300	24	24	20
2 200	25	25	4
6 000	23	23	9
10 000	16	16	10

【図 1 3 H】

事故金額マクロワード(／3)シソーラス

接頭語	L1ランク	L1セグメント	L2ランク	L2セグメント
0	{0,1,2}	{001,010,001}	{0,1}	{11,10}
1	{0,1,2}	{100,100,100}	{0,1}	{11,10}
2	{1}	{001}	{0}	{01}
6	{2}	{010}	{1}	{10}
10	{0}	{010}	{0}	{10}

【図 1 4 H】

事故金額マクロワード(／3)シソーラス

接頭語	L1冒頭	L1最終	L2冒頭
0	21	7	19
1	31	14	25
2	1	1	27
6	28	28	28
10	18	18	1

【図 1 4 A】

顧客名シソーラス

名前	L1冒頭	L1最終	L2冒頭
空	0	0	0
André	20	2	3
Ariane	5	6	12
Laure	8	12	14
Max	29	15	6
Oscar	26	26	7

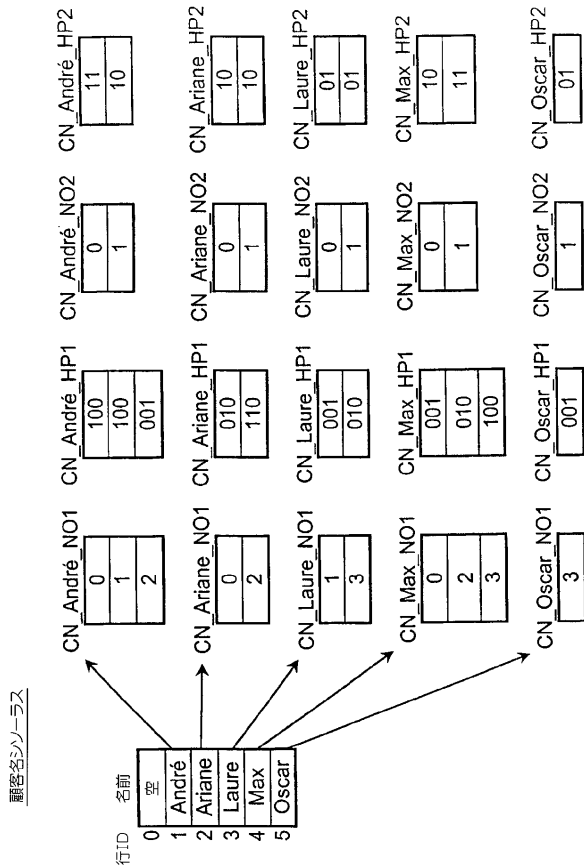
【図 1 5】

アドレス	L1ランク	次	L1セグメント
1	1	0	001
2	2	0	001
3	0	19	100
4	1	0	010
5	0	6	010
6	2	0	110
7	2	0	001
8	1	12	001
9	0	10	001
10	2	0	001
11	1	2	100
12	3	0	010
13	3	0	111
14	2	0	100
15	3	0	100
16	0	0	010
17	x	0	000
18	0	0	010
19	2	0	100
20	0	11	100
21	0	22	001
22	1	7	010
23	2	0	010
24	1	0	100
25	1	0	001
26	3	0	001
27	2	15	010
28	2	0	010
29	0	27	001
30	1	14	100
31	0	30	100
32	x	33	000
33	x	17	000

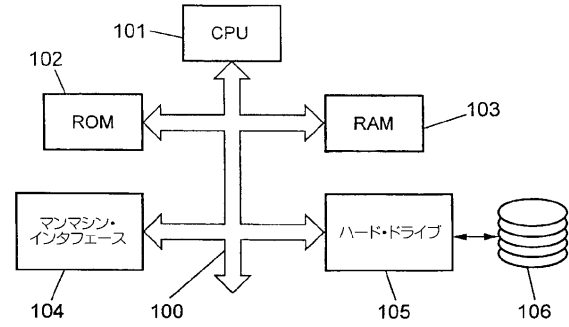
【図 1 6】

アドレス	L2ランク	次	L2セグメント
1	0	0	10
2	1	0	01
3	0	11	11
4	0	0	01
5	1	0	10
6	0	13	10
7	1	0	01
8	x	17	00
9	1	0	10
10	0	0	10
11	1	0	10
12	0	23	10
13	1	0	11
14	0	2	01
15	1	0	10
16	0	15	10
17	x	24	00
18	0	5	10
19	0	30	11
20	0	0	01
21	1	0	01
22	0	0	01
23	1	0	10
24	x	0	00
25	0	26	11
26	1	0	10
27	0	0	01
28	1	0	10
29	x	8	00
30	1	0	10

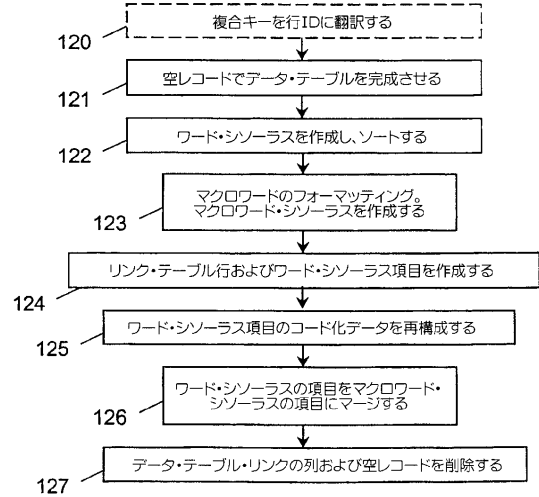
【図 17】



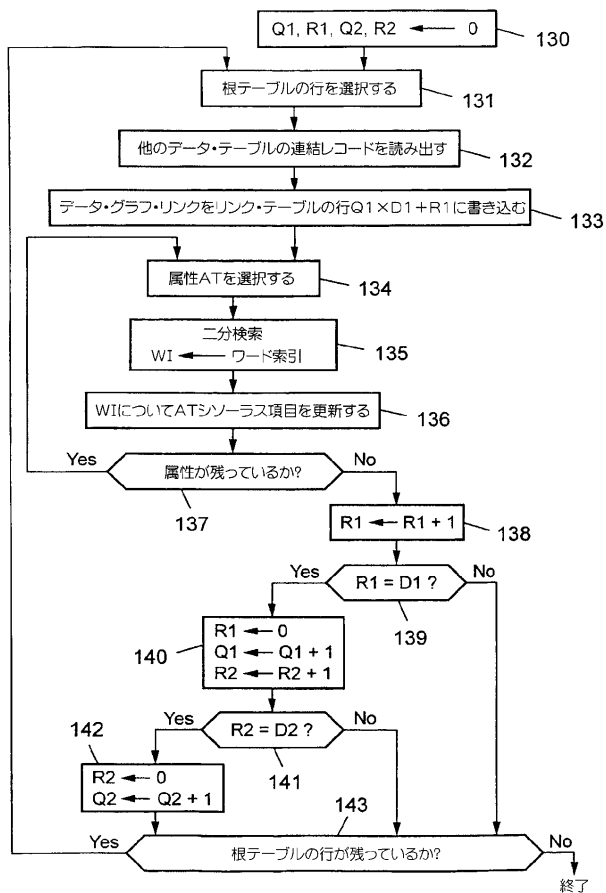
【図 18】



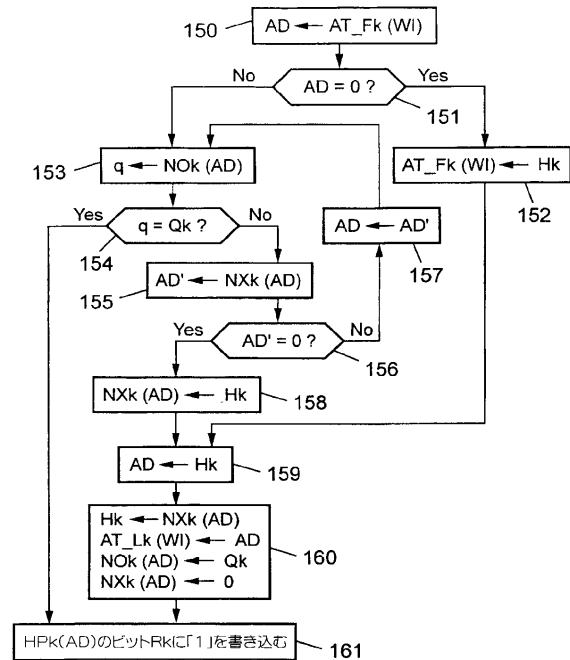
【図 19】



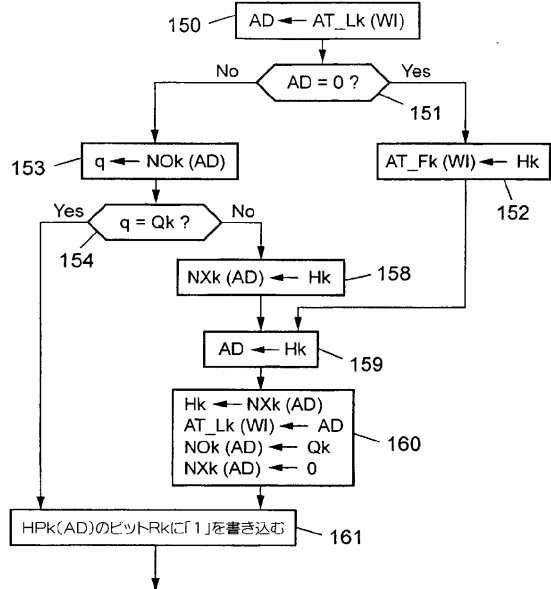
【図 20】



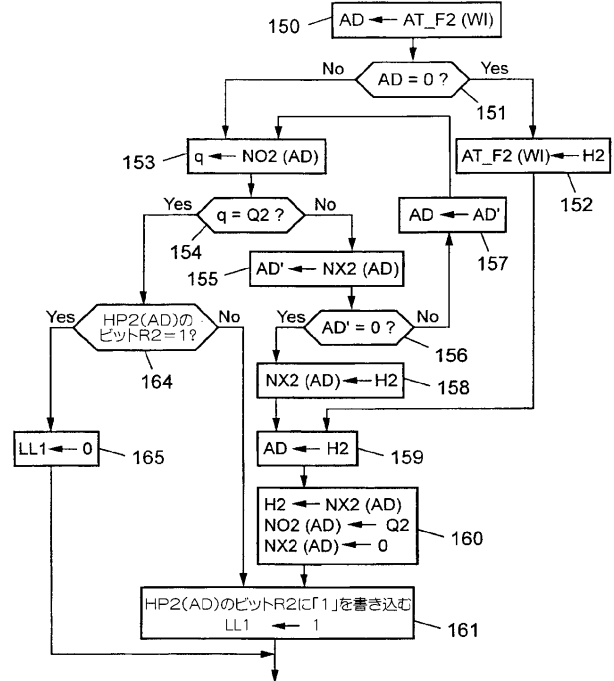
【図 21】



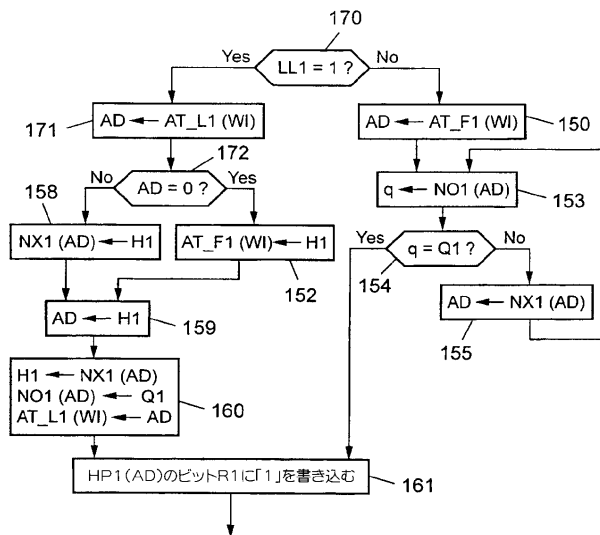
【図 2 2】



【図 2 3】



【図 2 4】



【図 2 6】

事故金額シソーラス

行ID	金額	ワード索引	フォーマット
0	空	0	1
1	300	1	1
2	800	2	1
3	1 000	3	1
4	1 300	4	1
5	2 200	5	1
6	6 000	6	1
7	10 000	7	1

【図 2 7】

事故金額マクロワード(／3)シソーラス

行ID	接頭語	マクロ索引	第1ワード	フォーマット
0	0	0	1	1
1	1	1	3	1
2	2	2	5	1
3	6	3	6	1
4	10	4	7	1

【図 2 5】

顧客名シソーラス

行ID	名前	ワード索引	フォーマット
0	空	0	1
1	André	1	1
2	Ariane	2	1
3	Laure	3	1
4	Max	4	1
5	Oscar	5	1

【図 2 8】

顧客名補助テーブル

行ID	CN_F1	CN_L1	CN_F2
0	0	0	0
1	1	3	1
2	4	5	3
3	6	7	5
4	8	10	7
5	11	11	9

【図 29】

事故金額補助テーブル

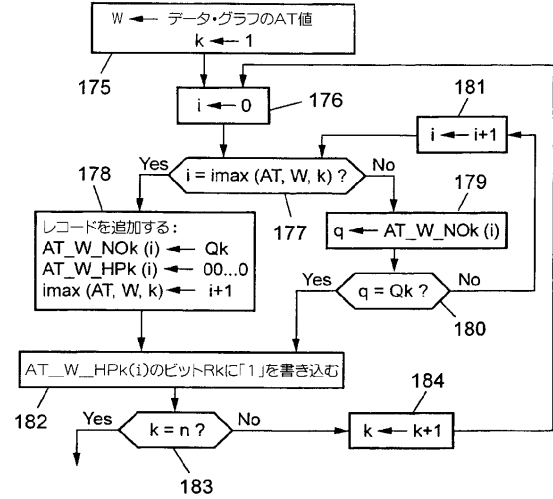
行ID	AA_F1	AA_L1	AA_F2
0	12	12	10
1	13	13	11
2	14	15	12
3	16	17	14
4	18	18	16
5	19	19	17
6	20	20	18
7	21	21	19

【図 30】

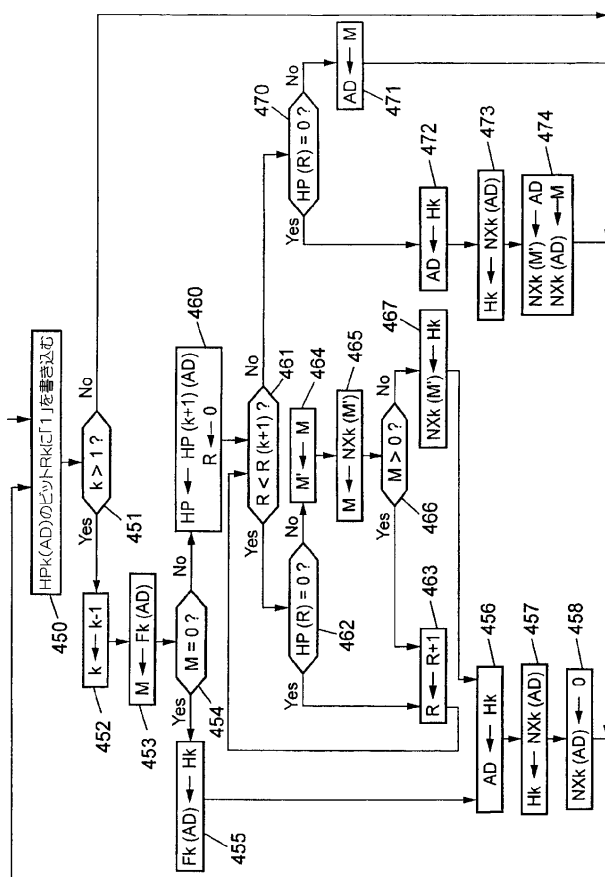
事故金額マクロワード(3)補助テーブル

行ID	AA_3_F1	AA_3_L1	AA_3_F2
0	22	24	20
1	25	27	22
2	28	28	24
3	29	29	25
4	30	30	26

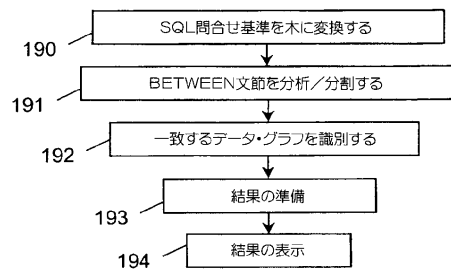
【図 33】



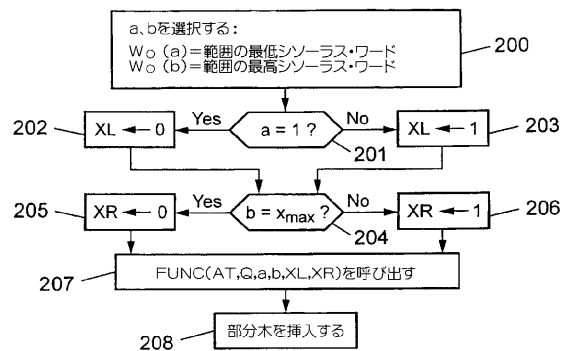
【図 35】



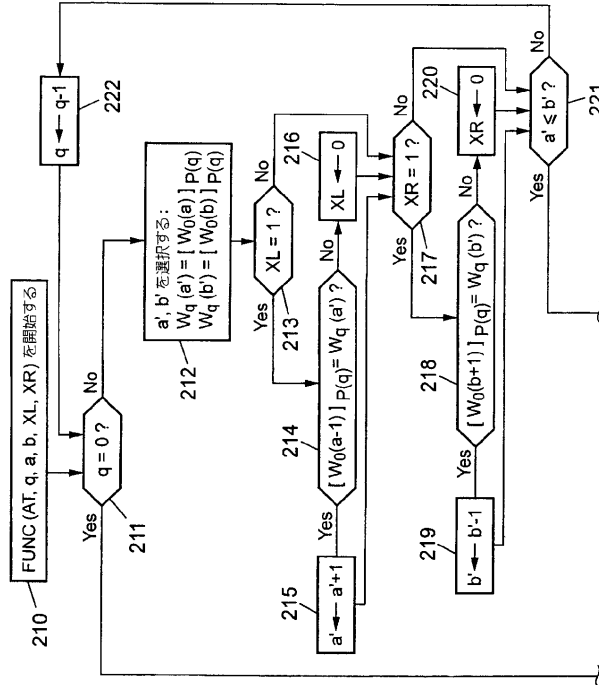
【図 36】



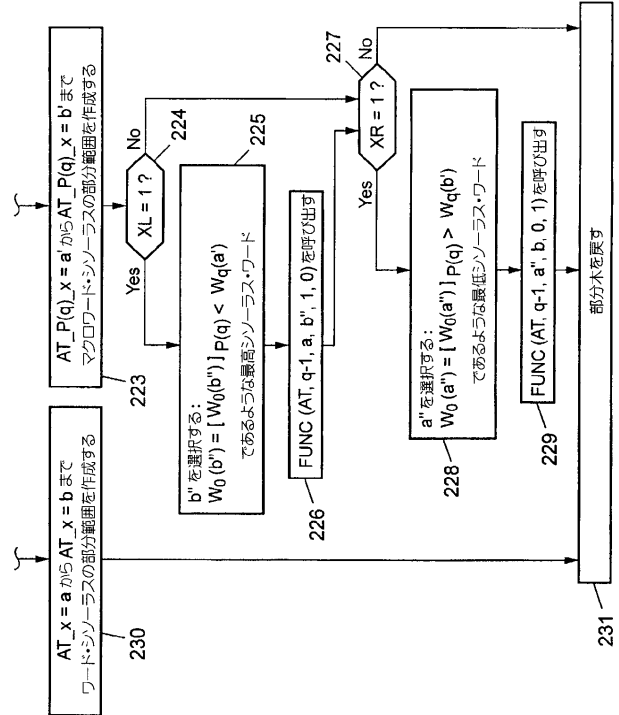
【図 39】



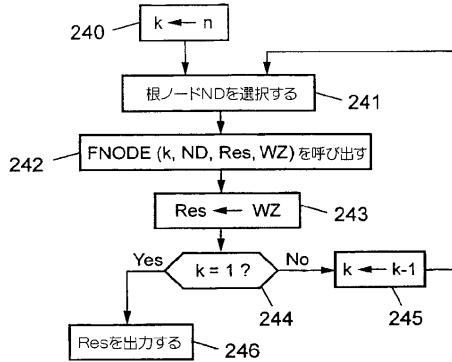
【図 40 A】



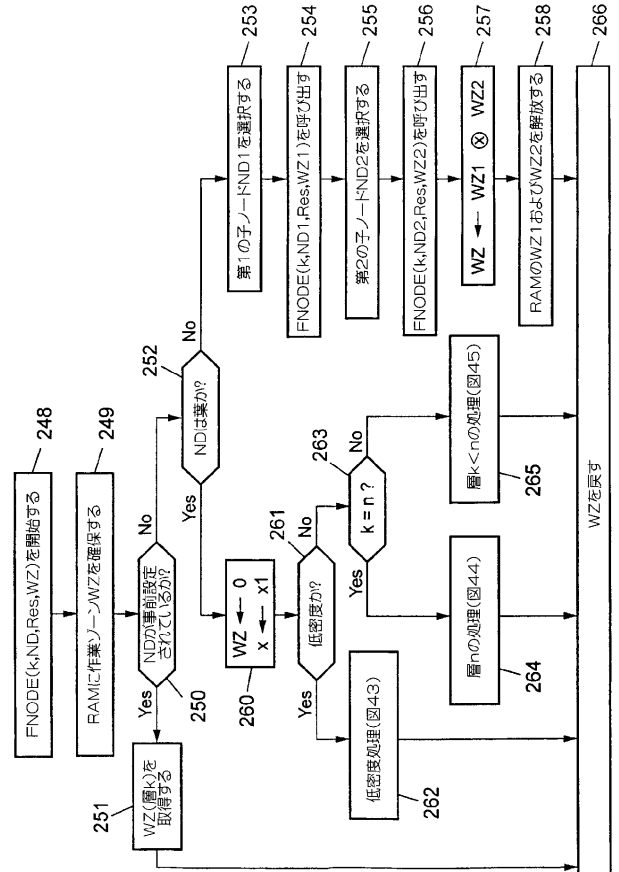
【図 40 B】



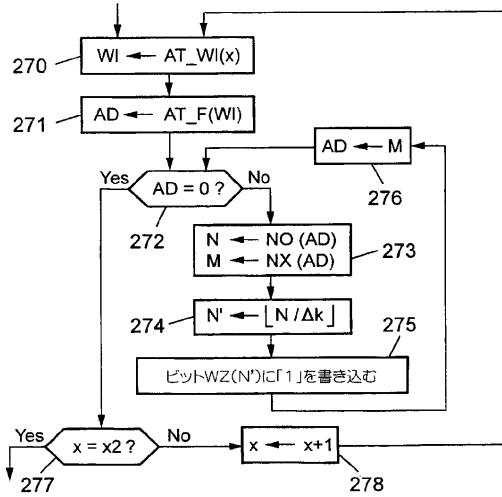
【図 41】



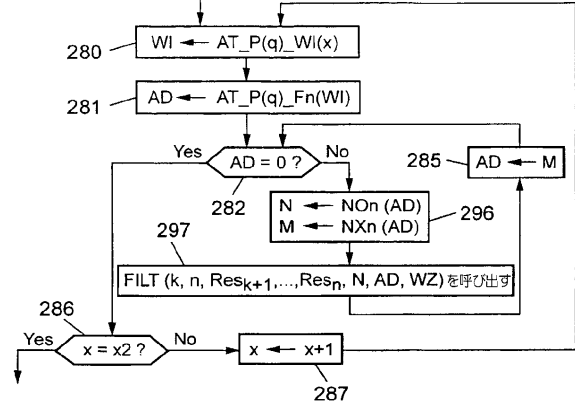
【図 42】



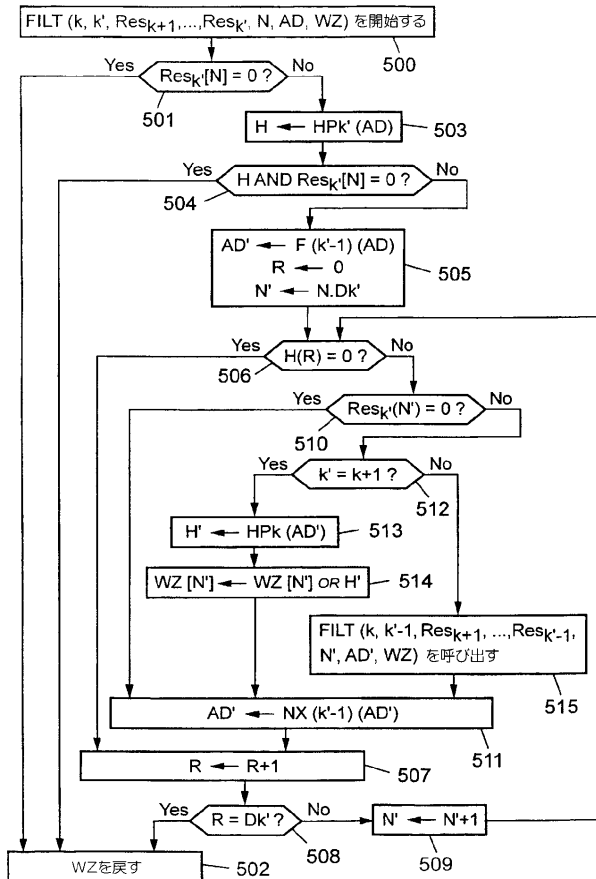
【図 4 3】



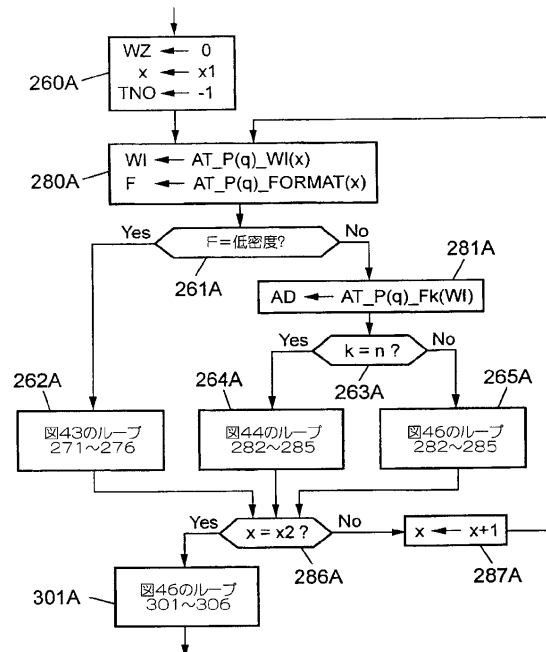
【図 4 7】



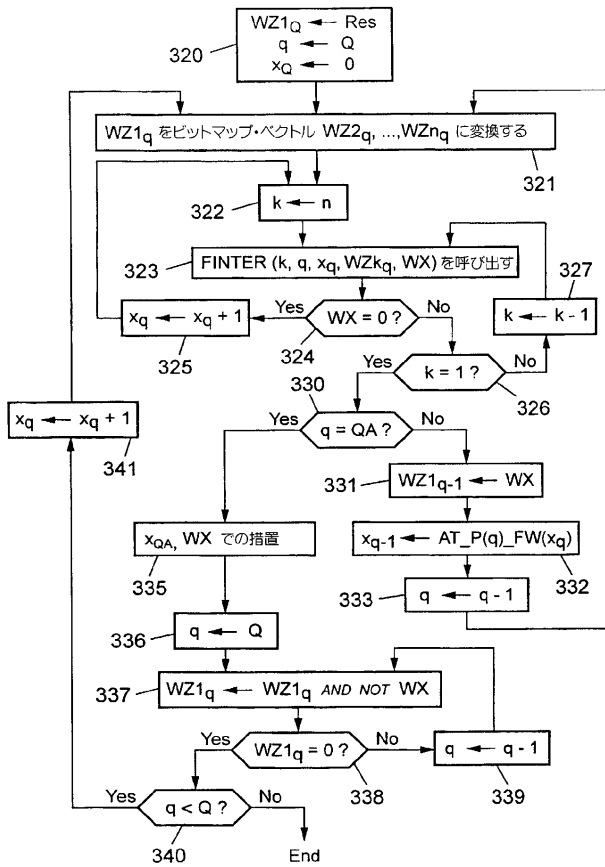
【図 4 8】



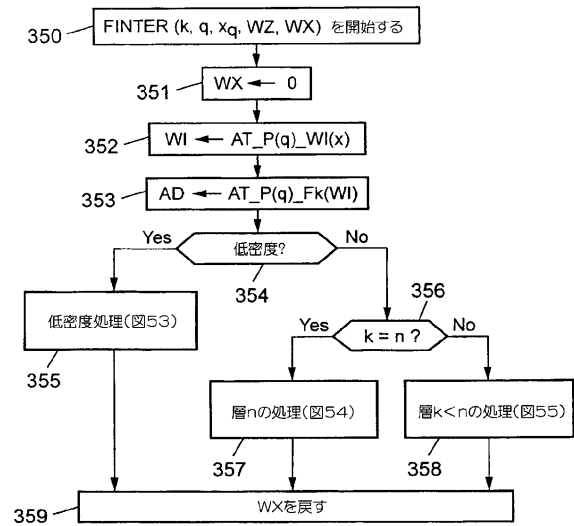
【図 5 0】



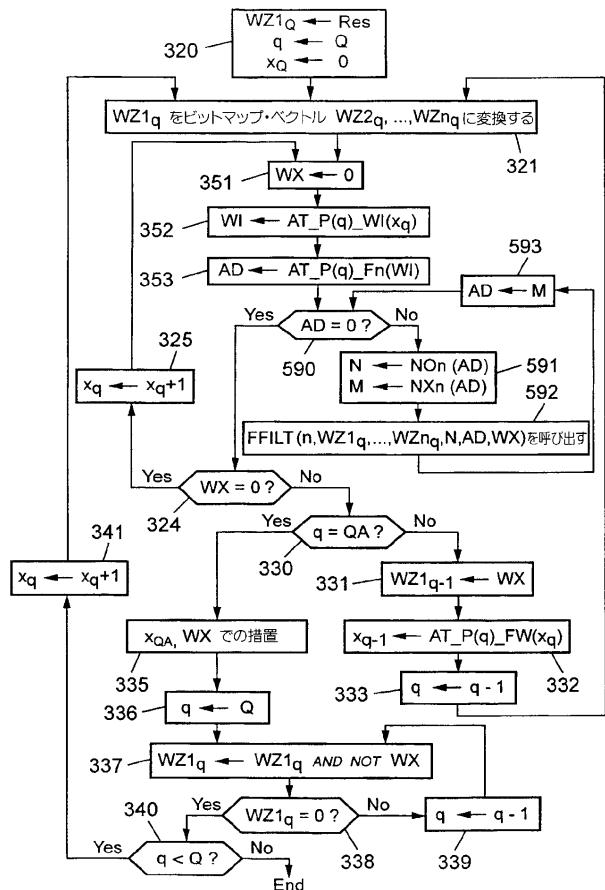
【図 5 1】



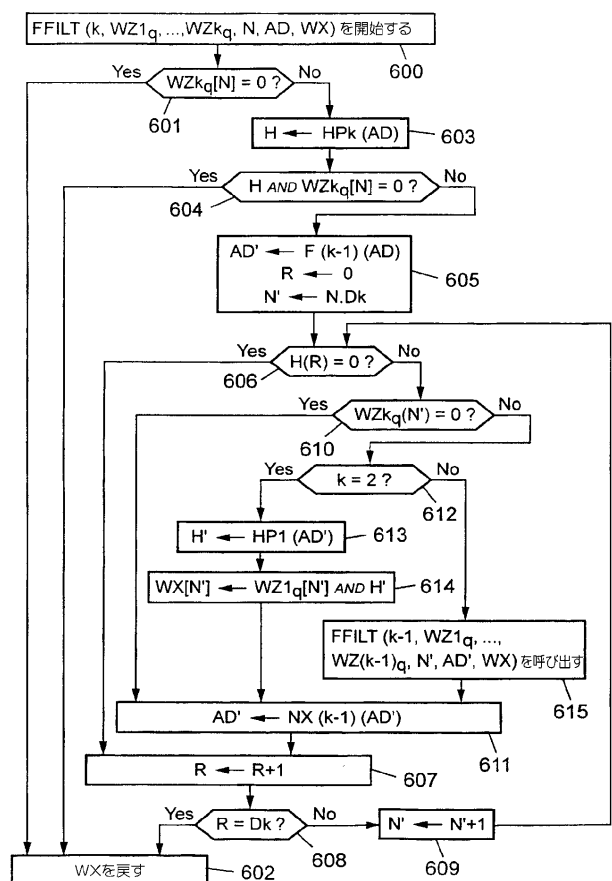
【図 5 2】



【図 5 6】



【図 5 7】



【図 5 8】

顧客名

NO2	F_AD'
0	1
1	5

【図 5 9】

事故金額

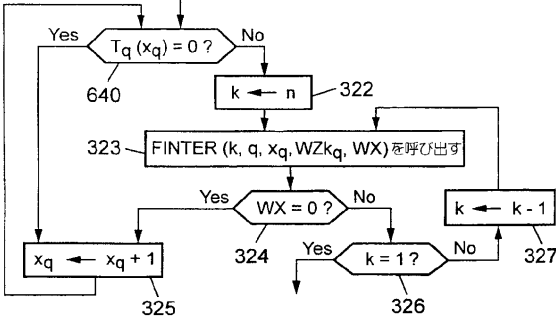
NO2	F_AD'
0	10
1	16

【図 6 0】

事故金額/3

NO2	F_AD'
0	20
1	24

【図 6 3】



【図 6 8】

出力テーブル

行ID	事故日	顧客名	保険日
0	1	1	2
1	2	4	4
2	3	1	2
3	8	1	2

索引

フラットファイル行ID	
0	0
1	/
2	1
3	2
4	/
5	/
6	/
7	/
8	3
9	/
10	/
11	/

【図 6 6】

出力テーブル

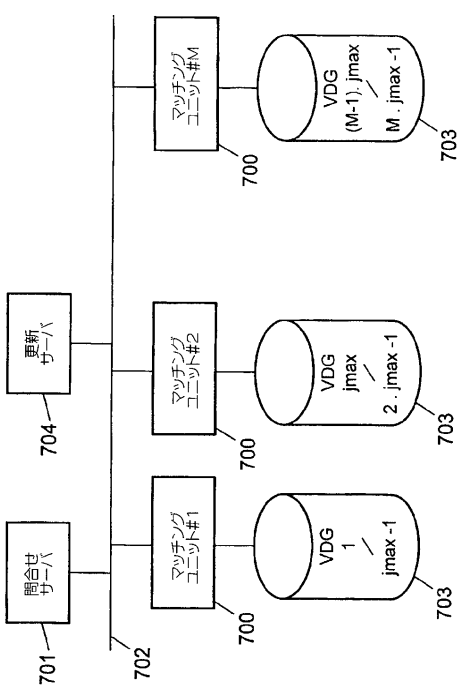
フラットファイル行ID	事故日	顧客名	保険日
0	1998 10 03	André	1998 09 09
1	/	/	/
2	1999 04 18	Max	1999 01 10
3	1999 06 12	André	1998 09 09
4	/	/	/
5	/	/	/
6	/	/	/
7	/	/	/
8	2000 03 20	André	1998 09 09
9	/	/	/
10	/	/	/
11	/	/	/

【図 6 7】

出力テーブル

フラットファイル行ID	事故日	顧客名	保険日
0	1	1	2
1	/	/	/
2	2	4	4
3	3	1	2
4	/	/	/
5	/	/	/
6	/	/	/
7	/	/	/
8	8	1	2
9	/	/	/
10	/	/	/
11	/	/	/

【図 7 2】



【国際公開パンフレット】

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau(43) International Publication Date
6 June 2002 (06.06.2002)

PCT

(10) International Publication Number
WO 02/44943 A2(51) International Patent Classification: **G06F 17/30**

(21) International Application Number: PCT/IB01/02792

(22) International Filing Date:
29 November 2001 (29.11.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

00403329.6	29 November 2000 (29.11.2000)	EP
00403330.4	29 November 2000 (29.11.2000)	EP
00403331.2	29 November 2000 (29.11.2000)	EP
00403332.0	29 November 2000 (29.11.2000)	EP

(81) Designated States (*national*): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZM, ZW.

(84) Designated States (*regional*): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

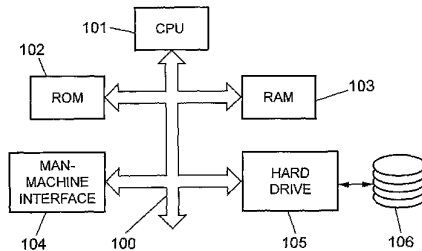
(71) Applicant: LAFAYETTE SOFTWARE INC. [US/US];
Five Palo Alto Square, 3000 El Camino Real, Palo Alto,
CA 94306 2155 (US).

(72) Inventor: KOSKAS, Elie, Ouzi; 9, Allée Eridan, F-95350
Saint Brice Sous Foret (FR).

Published:
— without international search report and to be republished
upon receipt of that report

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: METHODS OF ORGANIZING DATA AND PROCESSING QUERIES IN A DATABASE SYSTEM, AND DATABASE SYSTEM AND SOFTWARE PRODUCT FOR IMPLEMENTING SUCH METHODS



(57) Abstract: A reference table has columns associated with data attributes and rows containing related words assigned to those attributes in a collection of data, those words coming from different data tables having independent numbers of records. The stored data include word thesauruses associated with the attributes, and reference table row identifier lists respectively associated with thesaurus entries. Each word thesaurus associated with an attribute has a respective entry for each word assigned to this data attribute in the collection of data. The reference table, which may be a virtual table, defines a unified algebraic framework for the entries of all the thesauruses. Query criteria can be examined with reference to the relevant thesauruses to obtain a row-ID list or bitmap vector which represents all the reference table rows matching the query criteria, if any. The results can then be delivered through the original data tables, or preferably, by means of the thesauruses.

WO 02/44943 A2

WO 02/44943

PCT/IB01/02792

-1-

**METHODS OF ORGANIZING DATA AND PROCESSING QUERIES
IN A DATABASE SYSTEM, AND DATABASE SYSTEM AND SOFTWARE
PRODUCT FOR IMPLEMENTING SUCH METHODS**

BACKGROUND OF THE INVENTION

5 The present invention relates to relational database management systems (RDBMS), and more particularly to computerized systems for storing and accessing large amounts of data.

In a non-limiting manner, the invention is applicable to "data warehouses". On-line transaction processing (OLTP) systems, such as for bank teller transactions and airline reservations, are optimized for finding a record associated with a specific key, e.g. finding the information about employee 123124. By contrast, data warehouses are optimized for finding sets of records very quickly. The reason is that typical queries are of the form: "find all sales by region and quarter" or "find stores that sell the greatest volume of sportswear
10 per month" or "select the top 5 stores for each product category for the last year". Such queries must typically access large sets of rows in data tables. The query processing challenge is to process these queries without doing a linear scan of all or most of the database.

Five main approaches have been proposed to attack this problem: (i) multidimensional arrays; (ii) special indexes; (iii) table caching; (iv) optimized foreign key joins; and (v) approximation.

(i) Multidimensional arrays (i.e. matrices).

This strategy consists of implementing the data warehouse as a multidimensional array or matrix. Examples may be found in U.S. Patents No. 5,359,724 and No. 5,864,857. Each dimension corresponds to an attribute of the data. For example, a sales table can be viewed as a matrix with coordinates: store location, product type, customer id, and so on. A particular sale can be identified by specifying all of these attributes. The strategy works well for small databases or very dense ones. By dense, we mean that the
25 Cartesian product of possible values should all be meaningful, e.g., every customer is likely to buy every product from every store. Since this is rarely
30

WO 02/44943

PCT/IB01/02792

- 2 -

true, this scheme must be modified to deal with sparse values. This can be done by defining a notion of sparse attributes and dense ones. So, for example, it might be that every store carries every product (a dense relationship that can be stored in a matrix), but only some of these combinations are valid for any given customer. So, a conventional index would be used whenever customer sales are involved, but a dense one for queries involving store-wide or product-wide sales.

(ii) *Special indexes.*

Bitmap indexes are an index structure tailored to data warehouses (see, e.g. U.S. Patent No. 5,903,888). These indexes have already been used in some commercial products to speed up query processing. In its simplest form, a bitmap index on an attribute consists of one vector of bits (i.e. bitmap) per attribute value, where the size of each bitmap is equal to the number of records in the indexed relation. For example, if the attribute is day-of-week, then there would be seven bitmap vectors for that attribute, one for each day. The bitmap vector corresponding to Monday would have a 1 at position i if record i contains "Monday" in the day-of-week attribute. This single value-based approach is called a Value-List index. Other techniques (e.g. U.S. Patent No. 5,761,652) associate bit vectors with ranges of values, so there could, for a salary attribute, be a vector for the range 0 to 20,000 Euros, 20,000.01 to 35,000 Euros, and so on. Still others associate each bit vector with a bit value (a 1 or a 0) in a given position. So, if the attribute holds n bit numbers, then there would be $2n$ bit vectors (position 1, bit value 1; position 1, bit value 0; position 2 bit value 1; ...).

The benefit of bit vectors is that it is easy to use multiple bit vectors to answer a single query. Consider a query on several predicates, each of which is indexed. Most conventional database management systems would use just one of the indexes (the one that is most "selective" so returns the fewest rows), though some systems might attempt to intersect the record identifiers of multiple indexes.

Bitmaps work better, because they are more compact and intersecting several bitmaps is much faster than intersecting several collections of record

WO 02/44943

PCT/IB01/02792

- 3 -

identifiers. In the best case, the improvement is proportional to the word size of the machine. For example, suppose the word size is 32 bits. Then two bit vectors can be intersected 32 bits at a time. Each set of 32 bits corresponds to 32 record identifiers being intersected. That best case occurs when each
5 predicate is unselective (i.e. many records match each predicate value), but all the predicates together are quite selective. Consider for example the query: "Find people who have brown hair, glasses, ages between 30 and 40, blue eyes, work in the computer industry, live in California, ...".

So, matrices are best when sets of predicates are dense (all, or nearly
10 all, values in the Cartesian product are possible), bitmaps are best when predicates are neither dense nor individually selective. An intermediate approach (when there is insufficient density for matrices but many values in the Cartesian product are present) is to use multidimensional indexes. Multidimensional indexes such as quadtrees, R-trees and their successors are
15 implemented as variable sized grids on a multidimensional space. The grids are of variable sizes because the population of points differs in different places in a hyperspace. For intuition, consider a map of equi-population rectangles of France. The rectangles would be far more dense in Paris than in the alps. Indexes like this work well for spatial data (where they are used to find the
20 points contained in latitude-longitude quadrants). This alternative is little explored in the commercial arena except for geographical queries, however, because these schemes do not scale well with increasing dimensionality and commercial systems typically have far more than three dimensions.

(iii) Table Caching.

25 If one doesn't have the luxury to design new indexes on top of a database system (because one is not the implementer of that system) one can pre-compute a large number of anticipated aggregate queries and put them in tables. For example, if a large retailer frequently asks queries that sum the total sales across multiple stores or multiple products, one may store such
30 information in special tables. The main cost of such a strategy is maintaining these tables in the face of updates. (Disk space is no longer a major factor.) In the example, every sale of item I at store S would have to update the total

WO 02/44943

PCT/IB01/02792

- 4 -

product sales table for I and the total store sales table for S. So, this strategy is worthwhile if there are few updates between queries. The strategy is not worthwhile if there are many.

(iv) Optimized Foreign Key Joins.

5 Most queries in multidimensional tables entail joins between a central "fact table" (e.g. sales detail) and a set of dimension tables (e.g. store description, product description, customer description). These are known as "foreign key joins" since the customer identifier in the sales table, for example, is a key of the customer description table. (A key is a value belonging to an attribute such that only one record has that value in the attribute.) One way to
10 accelerate these joins is to create a linkage between fact table records and dimension records. This can be done in three basic ways

- (a) create an index that holds fact table record identifiers and dimension table record identifiers;
- 15 (b) create bidirectional pointers between fact table records and dimension table rows – this is what "object-oriented" databases do;
- (c) replace the customer record identifiers in the fact table by offsets into the dimension tables.

Choice (a) is the most independent of changes in the physical
20 organization of the tables and therefore is best for heavily updated systems, because changes to the dimension table can be reflected in the index to that table alone. Choice (b) is the least flexible to physical reorganization, because reorganizing a dimension table would entail updating the fact table. Choice (c) is a compromise of the two in that certain physical reorganizations can be done
25 to the dimension tables (e.g. changing its position on disk) without changing the fact table. Examples of join optimization may be found in U.S. Patents No. 5,548,754, No. 5,671,403, No. 5,724,568, No. 5,752,017, No. 5,761,657 and No. 5,822,747.

(v) Approximating the Result

30 Since most people use data warehouses to get strategic aggregate information, many would be happy with a fast approximation as long as it has error bounds. Typical work in this area is illustrated by U.S. Patent No.

WO 02/44943

PCT/IB01/02792

- 5 -

5,870,752, which shows how to estimate aggregate results in data warehouses while giving error bounds. The basic problem is that sampling all tables and then doing aggregates does not work in general. For example, if one wants to join R and S on their keys, then taking a 1/10 sample of each will give a size that is 1/100 of the size of the real join if the samples are random. So, one must be more clever. The idea is to take an initial set of tables R, S, T, ... that are linked by foreign key joins. Suppose for example that R is the fact table and the others are dimension tables. Take a sample of R and then perform all these foreign key joins based on the sample giving R', S', T', ... Now, if a query involves R, S, T and includes the foreign key links among these, then the query can be done with great accuracy on R', S', T'. The error can be estimated by considering the result obtained by several partitions of R' and looking at their variance.

An object of the present invention is to propose an alternative method of organizing a database management system, which enables an efficient query processing.

SUMMARY OF THE INVENTION

The invention proposes methods of organizing information and of processing queries in a database system, as set out in the appended claims 1 through 248. The invention further proposes a database system for managing information from a collection of data, comprising means arranged and programmed to implement a such method, as well as computer program products having instructions for carrying out such method.

Another aspect of the present invention relates to methods of handling integer lists in computer systems. In a non-limiting manner, this aspect of the invention is applicable in the RDBMS field, where the integer lists may represent identifiers of records in various tables.

It is well known that, in computer systems, integer lists may equivalently be stored and handled in the explicit form of integer lists or in the form of bitmap vectors. A bitmap vector has binary components each indicating whether an integer corresponding to the rank of the component belongs (1) or

WO 02/44943

PCT/IB01/02792

- 6 -

does not belong (0) to the list. The dimension of the vector has to be at least equal to the largest integer of the list.

The bitmap representation is convenient because a variety of manipulations can be performed on the coded lists by subjecting the binary components of the vectors to Boolean operations, which are the most basic operations in the usual processors. For example integer lists are readily intersected by means of the Boolean AND operation, merged by means of the Boolean OR operation, complemented by means of the Boolean NOT operation, etc.

When the integers of the lists are potentially big, the dimension of the bitmap vectors becomes large, so that the memory space required to store the lists in that form becomes a problem. When the lists are scarcely filled with integers of the big range, the explicit integer format is much more compact: a list of K integers in the range $[0, 2^{32}[$ requires $K \times 32$ bits vs. $2^{32} \approx 4.3$ billion bits in the bitmap format.

Bitmap compression methods have been proposed to overcome this limitation of the bitmap representation. These methods consist in locating regions of the vectors whose components have a constant value, so as to encode only the boundaries of those regions. The remaining regions can be coded as bitmap segments. An appreciable gain is achieved when very large constant regions are found. Examples of such bitmap compression methods as disclosed in US Patents No. 5,363,098 and 5,907,297.

This type of bitmap compression optimizes the storage of the encoded integer lists, but not their handling. Multiple comparisons are required to detect overlapping bitmap segments when performing basic Boolean operation on the bitmaps (see US Patent No. 6,141,656). This is not computationally efficient. In addition, when the coding data of the constant regions and bitmap segments are stored in memory devices such as hard drives (i.e. not in RAM), numerous disc read operations are normally required, which is detrimental to the processing speed.

An object of this aspect of the present invention is to propose alternative methods of encoding and/or combining integer lists, whereby lists of

WO 02/44943

PCT/IB01/02792

- 7 -

potentially large dimension can be efficiently handled.

Accordingly, the invention proposes methods of encoding and combining integer lists in a computer system, as set out in the appended claims 249 through 291. The invention further proposes a computer system, comprising means arranged and programmed to implement a such method, as well as computer program products having instructions for carrying out such method.

BRIEF DESCRIPTION OF THE DRAWINGS

Figures 1-3 show an example of data structure as typically used in a conventional relational database system.

Figure 4 is a diagram representing a data table tree in the example of figures 1-3.

Figures 5-7 are diagrams showing respective data graphs constructed with the tree of figure 4 and the data of figures 1-3.

Figure 8 is a flat file representation of the data tables of figures 1-3.

Figure 9 shows a link table as used in an embodiment of the invention.

Figures 10A-H show the contents of thesauruses corresponding to the data tables of figures 1-3.

Figures 11A-14A, 11G-14G and 11H-14H show other representations of the thesauruses of figures 10A, 10G and 10H, respectively.

Figures 15-16 illustrate the data stored in a data container in connection with the thesauruses of figures 14A, 14G and 14H.

Figure 17 shows another possible structure of the thesaurus of figures 10A-14A.

Figure 18 is a block diagram of a computer system suitable for implementing the invention.

Figure 19 is a flow chart showing a data graph creation procedure in accordance with an embodiment the invention.

Figure 20 is a flow chart showing a procedure applicable in stage 124 of figure 19.

Figures 21 and 22 are flow charts showing procedures applicable in step 136 of figure 20.

WO 02/44943

PCT/IB01/02792

- 8 -

Figures 23 and 24 are flow charts showing another procedure applicable in step 136 of figure 20 in two successive coding layers.

Figures 25-32 are tables showing a way of storing thesauruses constructed from the example of figures 1-3.

5 Figure 33 is a flow chart showing an alternative way of executing steps 135 and 136 of figure 20 when the thesauruses are stored as shown in figure 17.

Figures 34A and 34B are tables showing an alternative embodiment of the tables of figures 31-32.

10 Figures 34C and 34D are another representation of the tables of figures 34A and 34B.

Figure 35 is a flow chart showing a procedure applicable in the management of tables of the type shown in figures 34A and 34B.

15 Figure 36 is a general flow chart of a query processing procedure in accordance with an embodiment of the invention.

Figure 37 is a diagram showing an example of query tree referring to the example of figures 1-3.

Figure 38 is another diagram showing an expanded query tree obtained by analyzing the query tree of figure 37.

20 Figure 39 is a flow chart showing a procedure of analyzing the query tree.

Figure 40, which is obtained by placing figure 40A above figure 40B, is a flow chart of a recursive function referred in the procedure of figure 39.

25 Figure 41 is the flow chart procedure for identifying matching data graphs based on an expanded query tree as illustrated in figure 38.

Figure 42 is a flow chart of a recursive function FNODE called to in the procedure of figure 41.

Figures 43-45 are flow charts illustrating procedures executed in steps 262, 264 and 265 of figure 42, respectively.

30 Figure 46 is a flow chart showing an alternative embodiment of the procedure of step 265 of figure 42.

Figure 47 is a flow chart showing another alternative embodiment of the procedure of step 265 of figure 42, when the thesauruses are stored as

WO 02/44943

PCT/IB01/02792

- 9 -

illustrated in figures 34A and 34B.

Figure 48 is a flow chart of a recursive function FILT called in the procedure of figure 47.

Figure 49 is a flow chart showing another alternative embodiment of the procedure of step 265 of figure 42, when the thesauruses are stored as illustrated in figure 17.

Figure 50 is a flow chart of a variant of a leaf processing used in the function of figure 42.

Figure 51 is a flow chart showing a procedure applicable for scanning the thesaurus relating to a given attribute in order to retrieve the attribute values relevant to a database query.

Figure 52 is a flow chart of a function FINTER referred to in the procedure of figure 51.

Figures 53-55 are flow charts showing procedures executed in steps 355, 357 and 358 of figure 52, respectively.

Figure 56 is a flow chart showing an alternative procedure applicable for scanning the thesaurus relating to a given attribute in order to retrieve the attribute values relevant to a database query, when the thesauruses are stored as illustrated in figures 34A and 34B.

Figure 57 is a flow chart of a recursive function FFILT called in the procedure of figure 56.

Figures 58-61 show tables which may be stored to cooperate with the tables of figures 25-34.

Figure 62 is a flow chart showing a pre-filtering procedure which may be used prior to a thesaurus scanning similar to that of figure 51.

Figure 63 is a flow chart showing a part of a thesaurus scanning procedure according to figure 51, adapted to take into account a pre-filtering according to figure 62.

Figure 64 is a flow chart showing an alternative procedure applicable in step 358 of figure 52, when the thesauruses are stored as illustrated in figure 17.

Figure 65 is a flow chart showing a procedure applicable in step 335 of figure 51.

WO 02/44943

PCT/IB01/02792

- 10 -

Figures 66 and 67 show the contents of an exemplary output table used to provide a query response.

Figure 68 is a diagram illustrating another possible structure of the output table.

5 Figures 69 and 70 are flow charts showing procedures applicable in step 335 of figure 51 to construct an output table of the type shown in figure 68.

Figure 71 is a flow chart showing a procedure applicable in step 335 of figure 51 to perform computations in a database system by means of a computation table.

10 Figure 72 is a block diagram of another computer system suitable for implementing the invention.

DESCRIPTION OF PREFERRED EMBODIMENTS

VIRTUAL DATA GRAPHS

15 Figures 1-3 illustrate a collection of data which can be stored in a computer memory coupled with a processor arranged for running relational database management programs. This example will be referred to in the following description to give an illustration of the principles and embodiments of the invention where appropriate.

20 Figures 1-3 show a conventional type of data organization in a database system. The illustrated system handles data relevant to a hypothetical insurance company which manages policies for its clients. The data are organized in three tables relating to the clients, policies and accidents as shown in figures 1-3, respectively.

25 From a logical point of view, each data table consists of a two-dimensional matrix, with rows corresponding to respective records in the table and columns corresponding to respective data attributes of the records or structural features of the database (the latter type of column typically contains either local record identification keys or foreign keys designating records in a target table).

30 It will be appreciated, however, that for large databases the actual

WO 02/44943

PCT/IB01/02792

- 11 -

storage of the data in a memory medium, e.g. a magnetic disc, is frequently performed otherwise: each row typically has a memory address where the corresponding attribute values or keys are stored in the order of the columns and separated by predetermined symbols such as the encoded character "\".

5 In our simplified example given to facilitate the explanation of the proposed data structures, the tables are of modest size. In practice, there are usually more tables and more attributes (columns) per table (notwithstanding, one or more tables could also have a single column). Moreover, the data tables generally include much more records, up to thousands or millions of
10 rows depending on the application.

In that example, the database a group of seven attributes distributed into three sub-groups corresponding to the three data tables. Each attribute has a column in the data table corresponding to its sub-group. The client data table (figure 1) has three attributes, i.e. client name, birth year and gender. The
15 policy data table of figure 2 has two attributes, i.e. policy type ("car" or "house") and policy effect date, and a link column to the client table. The accident data table of figure 3 has two attributes, i.e. date of accident and amount of damages incurred in a given currency, and a link column to the policy table.

In a given data table, each record/row has a unique identifier, referred
20 to as a row-ID. This identifier corresponds to the memory address where the record is stored, usually through a conversion table. It may be stored as an identification key in a column of the data table for the purposes of unique row identification, but this is not compulsory. In our example, the row-ID's are integer indexes starting from zero for each data table, and they are not stored
25 explicitly in a column of the table.

Some of the tables are linked together, as indicated in the last column of figures 2 and 3. Two tables are directly linked if one of them (source table) has a link column provided for containing foreign keys designating records of the other one (target table).

30 Those foreign keys, hereafter called links, reflect the hierarchy and organization of the data handled in the relational database system. In our example, each accident dealt with by the insurance company is related to a certain policy managed by the company, hence the policy links of figure 3.

WO 02/44943

PCT/IB01/02792

- 12 -

Each policy is for a particular client of the company, hence the client links of figure 2. It will be noted that some links may be optional. For example, some accidents may involve third parties and if there is a separate table for third parties, then each record of the accident table may have a link to the third party table.

5 Each link typically consists of a row-ID in the target data table. For instance, the accident stored as row-ID = 0 in the accident table of figure 3, which took place on October 3, 1998 for an amount of 1,000 has a policy link pointing to the policy stored as row-ID = 1 in the policy table of figure 2, i.e. it relates to a car policy subscribed on September 9, 1998 by the client with
10 row-ID = 1 in the client table of figure 1, i.e. André, a man born in 1976. If the target table has other forms of record identification keys, for example compound keys, a link may also designate a target record as identified by such a key.

15 The construction of the links obeys a number of rules. In particular, the linked data tables have a directed acyclic graph structure such as a hierarchical tree organization illustrated in figure 4. A root table is defined as a data table for which no other data table has links pointing to its rows, such as the accident table of figure 3. In other words, a root table does not constitute a target table.
20 Likewise, a leaf table is defined as a data table with no link column, such as the client table of figure 1. In other words, a leaf table does not constitute a source table. Figure 4 shows only one root table, but the tree structure of the tables may have multiple roots.

It may happen in certain cases that a group of related data tables
25 exhibit circular links (for example, the client table may have a link column to the accident data table to indicate the first, or last, accident undergone by each client). In such a case, the tree organization of the data tables is first restored by canceling one link of the circle. Which link should be cancelled is dictated by the semantics of the database (in the above example, the link from the client
30 table to the accident table will naturally be cancelled).

Paths are defined in the data table tree from the root table(s) to the leaf tables. Each path from a root table to a leaf table is defined by a link column of the root table pointing to the leaf table, or by a succession of link columns via

WO 02/44943

PCT/IB01/02792

- 13 -

one or several intermediate tables.

In figure 4, two leaf tables have been added (dashed lines) to show a tree structure with multiple branching (the simplified example of figures 1-3 provides a tree with a single path shown with a solid line). The added leaf tables are a third party table as mentioned previously and a broker table which is a target table from the policy table, to contain data about the brokers who commercialize the policies.

The data table records that are linked together can be viewed in a similar tree representation (figures 5-7). The record tree of figure 5 shows that the accident #6 was related to policy #0 (car) subscribed by client #2 (Ariane) through broker #Y and involved third party #X. The solid lines represent respective links from the data tables of figures 2 and 3.

The record tree of figure 6 further shows a *Null* record which may be added in the accident table with a link to row-ID = 2 in the policy table, for the reason that, as apparent from the last column of figure 3, no accident has occurred under policy #2 (subscribed by client #4 (Max) for his house).

A *Null*, or dummy, record stands for the absence of data. All its attribute values are default values (*Null*), which means "no value". The purpose of inserting such dummy records in the present scheme is to make sure that any valid record in any data table belongs to at least one record tree stemming from a record of a root table (figure 4).

A *Null* record may also be present in each data table which is a target table for at least one link column of a source table. When a row of the source table has no foreign key in the corresponding link column, the record tree(s) including that row is (are) completed with a *Null* at the location of said target table. This situation occurs for the broker table in the example illustrated in figure 6. To represent this, a default value (e.g. -1) can be written in the link column of the source table, whereby the *Null* record is implicitly present in the target table.

The *Null* records are inserted where appropriate in a process of scanning every single path in the data table tree from the leaf table of said path to the root table, i.e. downwardly in figure 4. When examining one source/target table pair in the scanning of a path, the target table row-ID values that do not

WO 02/44943

PCT/IB01/02792

- 14 -

occur in the relevant link column of the source table are first listed, and then for each missing row-ID value of the list, a new *Null* record is generated in the source table with said missing row-ID value in said link column.

If a *Null* record is thus inserted in a data table having several link
5 columns, the *Null* record receives the default value (-1) in any link column other than the one pertaining to the path being scanned, to indicate that the corresponding link is to a *Null* record in the target table. This situation occurs for the third party table in the example illustrated in figure 6.

Scanning the data table tree from the leaves to the root is important.
10 Otherwise, *Null* records containing links to other *Null* records in a target table might be overlooked. An example is shown in figure 7 which shows a record tree relating to client #0 (Oscar) who has no (more) policy: the accident table contains a *Null* record pointing to another *Null* record of the policy table which, in turn, points to client #0; the root of the record tree would not be in the root
15 (accident) table if the paths were scanned upwardly.

In a conventional database organization as shown in figures 1-3, the link keys are provided to optimize the memory usage. To illustrate this, reference may be made to the flat file shown in figure 8, which has exactly the same informational content as the three data tables of figures 1-3 (the third
20 party and broker tables are ignored in the sequel).

A flat file has a column for each one of the attributes (columns) of the data tables. For each complete record tree that can be constructed with the data table tree structure of figure 4, the flat file has a row which contains, in the relevant columns, the attribute values of all the records of said tree. The rows
25 of the flat file are referred to herein as data graphs. Each data graph is identified by a flat file row-ID shown in the left-hand portion of figure 8. The record trees of figures 5-7 are compact representations of the data graphs at row-ID's 6, 9 and 11, respectively.

Although the flat file representation is sometimes referred to the
30 literature, it is of little practical interest for databases of significant size. The reason is that it requires excessive redundancy in the data storage.

For example, in our small-sized case, André's birth year and gender, as well as the details of his car policy are written three times in the flat file

WO 02/44943

PCT/IB01/02792

- 15 -

(row-ID's 0, 3 and 8), whereas they are written only once, along with link values, when the storage is in the form of data tables as in figures 1-3. With databases of realistic size, such redundancy is not acceptable.

The database system according to the invention makes use of the flat file concept. However, it does not require the storage of the flat file as shown in figure 8, hence the concept of "virtual flat file" containing "virtual data graphs" (VDG). The term "virtual" refers to the fact that the flat file or data graphs need not be maintained explicitly in memory, although their data structure is used as a reference in the execution of the method.

In a particular embodiment of the invention, the flat file is reduced to a link table as shown in figure 9. Each row of the link table corresponds to a respective row of the flat file, i.e. to a record tree as shown in figures 5-7.

The columns of the link table respectively correspond to the data tables of figures 1-3. In other words, each column of the link table is associated with an attribute sub-group which is the sub-group of attributes allocated to the corresponding (target) data table. Each column of the link table contains link values (row-ID's) designating records of the corresponding target data table.

The row of the link table corresponding to a given data graph contains a default value (-1) in the column corresponding to any data table having a *Null* record in the record tree representing said data graph.

The data table row-ID's found in one row of the link table enable the retrieval of linked data from the data table, i.e. a data graph or part of it. All the links are represented in the link table. If one replaces the row-ID's stored in the columns of the link table of figure 9 by the attribute values stored in the identified rows of the respective data tables of figures 1-3, one recovers the flat file of figure 8.

The proposed system further uses word thesauruses (figure 10A-G) each associated with a respective column of one of the data tables, i.e. with one of the attributes.

In a preferred embodiment, there is one word thesaurus for each attribute used in the database system. However, if some attributes are known to be never or almost never used in the query criteria, then it is possible to dispense with the thesaurus for such attribute.

WO 02/44943

PCT/IB01/02792

- 16 -

Each word thesaurus associated with one column of a data table has an entry for each attribute value found in that column. Such attribute value is referred to herein as a "word". A word has one entry in a thesaurus, and only one, as soon as it occurs at least once in the associated data table column.

5 The *Null* value is a valid word in the thesaurus.

The entries of each thesaurus are sorted on the basis of the attribute values. An order relationship is therefore defined for each attribute category. This requires attention when the attribute value fields of the thesaurus files are defined and dimensioned.

10 Typically, the words are in the ASCII format and their category is selected for each column among the categories "integer", "real" and "character string". Character strings are sorted according to the usual lexicographical order. A date field is preferably declared as a character string such as *yyyy (mm) (dd)* (figures 10B, 10E and 10F), *yyyy* representing the year, *mm* the
15 month (optionally) and *dd* the day in the month (optionally). The thesaurus sorting thus puts any dates in the chronological order. If the attribute category is "integer", the numbers are aligned on the right-hand digit, in order to provide the natural order relationship among the integer data values. If the attribute category is "real", the numbers are aligned according to their whole parts, with
20 as many digits on the right as in the value having the longest decimal part in the column.

The *Null* value is at one end (e.g. at the beginning) of each sorted thesaurus.

Each entry $E(W)$ for a word W in a thesaurus associated with a column
25 $C(T)$ of a data table T contains information for identifying every row of the flat file which has the attribute value W in the column corresponding to $C(T)$. When the flat file is stored virtually in the form of a link table, the information contained in entry $E(W)$ is used for identifying every row of the link table which, in the column corresponding to the data table T , has a link pointing to a row
30 having the value W in column $C(T)$.

In other words, with the contents of the entry $E(W)$ in the thesaurus associated with column $C(T)$, we can retrieve all the data graphs whose corresponding attribute has the value W .

Such contents represent a row-ID list pointing to rows of the (virtual) flat file, i.e. a data graph identifier list. Such list may be empty, in particular for the *Null* value in some of the thesauruses (as in figures 10A-C).

Two alternative representations of the data graph identifier lists in the thesauruses are illustrated in figures 10A-G for the seven attribute columns of figures 1-3. The first one is the form of explicit integer lists.

The second (equivalent) representation is in the form of bitmap vectors whose length is equal to (or greater than) the number of rows in the virtual flat file, i.e. the number of data graphs. The bit of position i in a bitmap vector ($i \geq 0$) indicates whether the integer i belongs (1) or not (0) to the row-ID list represented by the bitmap vector. In our simplified example, the flat file has 12 rows so that the bitmap vectors may be of dimension 12.

The above-described data structure, comprising a virtual flat file and sorted thesaurus files pointing to rows of the virtual flat file is referred to herein as a VDG structure.

The VDG structure provides a powerful tool for efficiently processing queries in the database.

The virtual flat file is a reference table which defines a unified algebraic framework for the entries of all the thesauruses. The query criteria are examined with reference to the relevant thesauruses to obtain a flat file row-ID list (or bitmap vector) which represents all data graphs matching the query criteria, if any. The results can then be delivered by accessing the link table rows pointed to in that row-ID list to read the links which appear in part or all of the columns in order to retrieve attributes values as desired for the result presentation.

The processing with reference to the thesauruses mainly consists in logical operations performed on the row-ID lists to which they point. If they are represented as integer lists, such operations can be reduced to basic merge, intersect and/or complement operations, which respectively correspond to Boolean OR, AND, NOT operations in the bitmap representation.

The VDG structure also provides an efficient tool for accessing the contents of the database, which does not require accesses to the data tables. This tool is well suited to queries having special result presentation features

WO 02/44943

PCT/IB01/02792

- 18 -

such as SORT, COUNT, DISTINCT, ORDER BY, GROUP BY, etc. clauses, and also for carrying out any type of calculation on the data values of the records which match the query.

Example 1:

5 As an illustration, consider the following query: find the client name and accident date for all car accidents that incurred damages higher than 900, and group the results according to the client name. The query may be processed as follows. First, all the flat file row-ID lists identified in the accident amount thesaurus entries relating to amounts higher than 900 (the five last rows of figure 10G) are merged, which yields the list {0, 1, 3, 5, 6, 7} (or the bitmap vector 110101110000 obtained by a bitwise Boolean OR). Then the intersection of that list with the row-ID list identified in the policy type thesaurus entry relating to the value "car" (the second row of figure 10D) is determined. The result list {0, 3, 5, 6} (or bitmap vector 100101100000 obtained by a bitwise Boolean AND) specifies the data graphs that satisfy the query criteria. Finally, 15 the entries of the client name thesaurus (figure 10A) are read sequentially and when there is a non-empty intersection between the result list and the row-ID list identified in the client name thesaurus entry, the link table rows having their row-ID's in that intersection are read to retrieve the desired attribute values. In 20 our case, the output would be: André [accident dates 1998 10 03 (#0) and 1999 06 12 (#3)], Ariane [accident date 1999 12 09 (#6)] and Laure [accident date 1999 12 08 (#5)].

The above type of processing is applicable to any kind of query. The response is prepared by referring only to the sorted thesauruses, which 25 implicitly refer to the flat file framework. Once an output flat file row-ID list is obtained, the link table or the thesauruses can be used for retrieving the data of interest.

Example 2:

30 To further illustrate the outstanding performance of the VDG scheme, let us consider the query which consists in identifying any client who has had a car accident before the beginning of the civil year of his or her 35th birthday. In

WO 02/44943

PCT/IB01/02792

- 19 -

a typical conventional system, all the records of the accident data table of figure 3 have to be read to obtain the date attribute and policy link values. For each accident record, the policy data table is read at the row-ID found in the policy link column to obtain the policy type attribute and client link values and then, if the policy type is "car", another access to the client data table is necessary at the row-ID found in the client link column to obtain the birth year attribute value. The latter value is compared with the date attribute value previously obtained in the accident table to determine whether the criteria of the query are fulfilled.

If the data tables are sorted beforehand on the basis of the attributes referred to in the query criteria, such conventional processing may be accelerated by limiting the number of disc accesses. This requires data table sorting every time records are added, deleted or amended, which is not practical when the collection of data is large. And it is possible only in specific cases dictated by the data table sorting rule.

For example, if the client and policy tables were respectively sorted on the basis of the birth year and policy type attributes, the preceding request could be processed in a less prohibitive time by accessing the data records in a suitable order and with the help of the computer cache memory. However, the tip would not apply to other similar queries (e.g., assuming an additional column in the policy table for containing excess amounts, the identification of all accidents for which the damage amount was more than ten times the excess amount would raise the same problems).

With the VDG scheme, the above illustrative query can be dealt with in a very efficient manner. By means of the client birth year thesaurus (figure 10B) and the accident date thesaurus (figure 10G), the computer identifies the {client birth year, accident date} word pairs which satisfy the date criterion, i.e. accident date earlier than beginning of client's birth year + 35. This is done without worrying about whether the accident was undergone by the client. Such identification is relatively easy for any possible pair of attributes since any attribute likely to be referred to in queries has a sorted thesaurus. For each identified word pair, the intersection of the two flat file row-ID lists of the thesaurus entries is obtained. The resulting integer lists are merged. Then the computer intersects the row-ID list of the entry relating to the value "car" in the

WO 02/44943

PCT/IB01/02792

- 20 -

policy type thesaurus (second row in figure 10D) with the list
{0, 1, 3, 5, 6, 8, 10} resulting from the merger. The resulting list {0, 3, 5, 6, 8}
designates a set of matching rows in the link table, from which the relevant
client names (André – 3 times –, Laure and Ariane) are readily retrieved by
5 accessing the client table records whose row-ID's appear in the matching rows
and in the client column of the link table.

It is noted that, when processing a query, the link table is simply used
as a mean to retrieve the data of interest. Different ways of achieving this
retrieval function may be thought of.

10 A method is to keep the original data tables (figures 1-3) in memory.
However, it is worth noting that the link columns may be deleted from those
data tables, since their contents are already present in the link table.

From the observation that all possible attribute values are stored in the
corresponding thesauruses, another method is to store in the link table pointers
15 to the thesauruses. The latter method reduces the required disc space since an
attribute value has to be written only once, even if the value occurs frequently
in a data table column. It enables to quickly retrieve attribute values which
occur in a given flat file row without requiring the use of the original data tables.

For certain attributes, it may be interesting to store the explicit attribute
20 values in the link table, i.e. like in the flat file. In particular, this may be
interesting for numerical fields (usually of smaller size than character strings)
whose values are very dispersed and which are often requested among the
output attributes of a query response (e.g. money amounts). If those values are
explicitly written in the link table, there can be an appreciable gain in the disc
25 accesses required for fetching the output data, at the cost of a moderate
increase in the needed disc space.

In the foregoing explanations, the link table is a sort of skeleton of the
flat file, which is stored to facilitate the data retrieval once the list of flat file row-
ID's matching the query has been determined by means of the sorted
30 thesauruses.

Notwithstanding, storing a link table or any form of table reflecting the
flat file structure is not strictly necessary. In an advantageous embodiment, the

data graphs (or their portions requested for the result presentation) may be recovered from the thesaurus files only. To illustrate this, consider again Example 2. Once the result list {0, 3, 5, 6, 8} of matching virtual flat file rows has been obtained by processing the query criteria with reference to the thesaurus files, it is possible to scan the client name thesaurus and, for each word (client name), to intersect the flat file row-ID list represented in the thesaurus with the result list. If the intersection is non-empty, the word is included in the output. It may be accompanied with the intersection list to allow the user to quickly obtain further information from the relevant data graphs. This method requires the minimum memory space since only the thesaurus files need to be stored.

Even if a link table is stored, it may be advantageous, for certain queries, to retrieve the attribute values by scanning the thesaurus(es) as indicated hereabove rather than through the link table. This may occur, in particular, to perform computations on the data values when there is a relatively slow interface between the query processor and the data tables, e.g. an ODBC interface ("Open DataBase Connectivity").

Another advantage of the VDG scheme is that it provides a query processing engine which can co-exist with the data tables in their original form. Changes in the thesaurus entries are then done in response to corresponding changes in the original data tables. This is an interesting feature for users who find it important to keep their data in the form of conventional tables, because they do not want to be too dependent on a new system or because they need to access their tables through a conventional interface for other applications.

25 MACROWORDS

The above-described VDG's are advantageously completed with prefix thesauruses also referred to as macroword thesauruses.

Like the above-described word thesauruses, each macroword thesaurus is associated with one attribute, i.e. one column of one data table. In addition, it has a prefix length (or truncation length) parameter.

Each entry of the macroword thesaurus relates to a range of attribute values, and contains or points to data for identifying all the flat file rows having,

in the column corresponding to said attribute, an attribute value which falls within said range. The range corresponding to the entry of the macroword thesaurus corresponds to a prefix value having the prefix length assigned to the thesaurus: any word beginning by such prefix value has its flat file row-ID list included in that of the macroword. If the prefix length is noted P, a macroword $C_1C_2...C_P$ is the set of all values of the attribute which begin by the P characters or digits $C_1C_2...C_P$. The limit case where the prefix length is the number of characters or digits of the value field (i.e. truncation length is zero) is the word thesaurus described previously.

10 In other words, the macroword thesaurus entry identifies the flat file row-ID list (or bitmap vector) corresponding to the merger of the flat file row-ID lists (or to the logical OR between the bitmap vectors) which are identified in the entries of the word thesaurus corresponding to the individual words encompassed by the macroword.

15 Each thesaurus (word or macroword) associated with an attribute AT can thus be defined with reference to a partition into subsets of the set of words which can be assigned to attribute AT in the relevant data table record. It has a respective entry for each subset including at least one word assigned to attribute AT, this entry being associated with a flat file row-ID list including any
20 ID of a flat file row having a word of the subset assigned to attribute AT. In the case of a macroword thesaurus, the partition is such that each subset consists of words beginning by a common prefix. In the case of a word thesaurus, the partition is such that each subset consists of only one word.

As an example, figure 10H shows the accident amount macroword thesaurus for a truncation length of 3 characters. It is not necessary to repeat
25 the *Null* entry, which is already in the word thesaurus. Such a macroword thesaurus provides substantial economy in terms of disc accesses and flat file row-ID list mergers. For example, for obtaining information about the accidents that had an amount between 1,000 and 1,999, one access to the macroword thesaurus of figure 10H is enough to obtain the relevant list of flat file row-ID's
30 {0, 3, 6}, whereas it would require two thesaurus accesses and one merge operation with the non-truncated accident amount thesaurus of figure 10G. The

gain can be quite substantial for large databases and attributes of high cardinality, i.e. with many possible attribute values.

Macroword thesauruses based on prefix or truncation lengths provide a great flexibility in the processing of range-based query criteria. It is possible, for
5 a given attribute, to provide several macroword thesauruses having different prefix lengths in order to optimize the processing speed of various queries.

Typically, a date attribute may have a yearly macroword thesaurus (prefix length = 4) and a monthly thesaurus (prefix length = 6) in addition to the (daily) word thesaurus. Any other kind of attribute (numbers or text) may lend
10 itself to a convenient macroword thesaurus hierarchy.

VDG COMPRESSION

With the VDG scheme as described so far, the memory space required by the thesaurus files is not optimized.

The row-ID's being integers typically coded with 32 bits, if a word
15 occurs n times in the attribute column of the flat file of figure 8, $n \times 32$ bits are needed to explicitly encode its flat file row-ID lists. If the flat file has $N_{\max}$ rows (for example millions of rows), $N_{\max}$ bits are needed for each entry in the bitmap representation, for whatever value of n .

Generally speaking, for an attribute of high cardinality, such as the date
20 or amount attributes (figures 10E-G), the flat file row-ID lists are scarcely filled, so that the explicit integer list representation is satisfactory in terms of memory requirement, while the bitmap representation can be prohibitive for large flat files. Other attributes have a low cardinality, such as the client gender or policy type attribute in our example (figures 10C-D), whereby the bitmap
25 representation is well suited, while the integer list representation is unfavorable.

It is possible to adopt for each thesaurus a representation which is believed to be the most appropriate in order to reduce the needed memory space. However, this requires an a priori knowledge of how the attribute values will be distributed. Many attributes can be ambiguous in this respect, and the
30 optimization may also be difficult for different macroword sizes relating to a given attribute.

Bitmap compression methods as known in the art (e.g. US Patent No. 5,363,098 or No. 5,907,297) may also be used. A problem is that those methods are designed essentially for optimizing the storage volume, not the processing speed. In the VDG context, the advantage of reduced memory space may be counterbalanced by the disadvantage of longer response times due to multiple compression and/or decompression operations when processing a query. To the contrary, it is desired to increase the processing speed as much as possible.

In the preferred implementation of the VDG scheme, the compression of the flat file row-ID lists in the thesauruses is carried out by dividing a range covering all the row-IDs of the flat file into subsets according to a predetermined pattern. Then, each flat file row-ID list of a thesaurus entry is encoded with data for locating in the pattern each subset of the range which contains at least one row-ID of the list, and data representing the position of each integer of the row-ID list within any subset thus located.

The row-ID range $[0, N_{\max}]$ is selected to be equal to or larger than the number of rows in the flat file. The "predetermined pattern" conveniently defines the "subsets" as consecutive intervals $[0, D1-1]$, $[D1, 2 \times D1-1]$, etc., having the same length $D1$ within said range.

The coding data can then be produced very simply by Euclidean division. For any positive numbers x and y , we note $\lfloor x \rfloor$ the integer equal to or immediately below x , $\lceil x \rceil$ the integer equal to or immediately above x , and $x \bmod y = x - \lfloor x/y \rfloor \cdot y$. A Euclidean division by $D1$ is performed for each row-ID N of the input list. The quotient $Q1 = \lfloor N/D1 \rfloor$ indicates the rank of the corresponding interval in the pattern ($Q1 \geq 0$), while the remainder $R1 = N \bmod D1$ represents the position of the row-ID within the interval ($0 \leq R1 < D1$). The decoding is also very simple: from the encoding data $Q1$ and $R1$ for an item of the coded list, the row-ID is $N = Q1 \times D1 + R1$.

Advantageously, the interval length is a whole power of 2, so that the Euclidean divisions are performed by straightforward bit shift operations. A typical length is $D1 = 2^7 = 128$.

The encoding method can be expressed equivalently by referring to the

WO 02/44943

PCT/IB01/02792

- 25 -

bitmap representation. Each bitmap vector is divided into bitmap segments (or other types of bit groups if a more tortuous pattern is referred to), and for each segment containing at least one "1", the coding data include the rank (= Q1) and the contents of the segment. The all zero segments are discarded.

5 Figures 11A, 11G and 11H are other presentations of the client name and accident amount word thesauruses of figures 10A and 10G and of the accident amount macroword thesaurus of figure 10H, with $D1 = 3$ (a non-typical value of $D1$ used here for conciseness). The second columns are copied from the last columns of figures 10A, 10G and 10H, respectively, with blanks to
10 highlight the segmentation of the bitmap vectors. The third columns show the lists of ranks (= Euclidean quotients $Q1$) resulting from the encoding, and the fourth columns show the corresponding non-zero bitmap segments (having a 1 at the position of each remainder $R1$).

It is observed that for each thesaurus entry, the ranks $Q1$ form an
15 integer list included in the range $[0, N1_{\max}]$, with $N1_{\max} = \lceil N_{\max}/D1 \rceil$.

According to a preferred embodiment of the invention, a similar type of encoding can be applied to those rank lists. The encoding process may be iterated several times, with the same encoding pattern or different ones. In particular, the interval lengths could vary from one iteration to the next one.
20 They are preferably whole powers of 2.

The ranks and bitmap segments obtained in the first iteration with the interval length $D1$ are called layer 1 (or $L1$) ranks and layer 1 segments (figures 11A, 11G and 11H). Those obtained in the second iteration, with an interval length noted $D2$, are called layer 2 (or $L2$) ranks and layer 2 segments (figures
25 12A, 12G and 12H), and so forth.

In the following, n denotes the number of encoding layers numbered k with $1 \leq k \leq n$, layer k having a divisor parameter Dk , and the product

$$\Delta k = \prod_{k'=1}^{k-1} Dk' \text{ being the number of flat file row-ID's encompassed by one bit of a}$$

layer k bitmap segment ($\Delta 1 = 1$).

30 In the simplified case illustrated in figures 12A, 12G and 12H, $n = 2$ and the second encoding layer uses $D2 = 2$. The columns labeled "L1 Bitmap" are a

bitmap representation of the layer 1 rank lists, with blanks to highlight the further bitmap segmentation leading to the layer 2 data shown in the last two columns.

5 The layer 1 and layer 2 coding data are summarized in figures 13A, 13G and 13H which show a possible way of storing the flat file row-ID list information. It is noted that storage of the layer 1 rank lists is not strictly necessary since those list are completely defined by the layer 2 data. However, it will be appreciated further on that such storage somewhat simplifies the query processing in certain embodiments of the invention.

10 The same kind of encoding may be used for any one of the word and macroword thesauruses. However, it is also possible for some of them to retain a conventional type of row-ID list storage (explicit integer lists or bitmap vector), i.e. $n = 0$. In particular, the explicit integer list representation may remain well-suited for scarcely distributed thesauruses.

15 Figures 14-16 show another possible way of storing the information contained in the thesauruses of figures 13A, 13G and 13H. For each encoding layer, the thesaurus entries are associated with respective chains of records in a data container (figure 15 for layer 1 and figure 16 for layer 2) including a rank file and a bitmap segment file. Each record in the layer k rank file ($1 \leq k \leq n$)
20 has a field for receiving a rank value (between 0 and $N_{k_{\max}}-1$) and a field for receiving an address of a next record in the rank file. A default value in the next address field (0 in the example shown) means that the record is the last one of the chain. The bitmap segment file (right-hand parts of figures 15 and 16) is addressed in the same manner as the associated rank file. In each record for
25 layer k , its has a bitmap field of D_k bits for receiving the bitmap segment associated with the rank stored in the corresponding record of the rank file. It will be appreciated that the rank values and next record addresses could also be stored in two separated files having a common addressing rather than in two fields of the same file.

30 For each VDG coding layer k , an entry in a thesaurus has a head address field for containing an address in the layer k rank file where a first rank record concerning the entry is stored. From there, the relevant rank chain can

WO 02/44943

PCT/IB01/02792

- 27 -

be retrieved. For example, Max's layer 1 ranks 0, 2 and 3 (figure 13A) are retrieved by accessing the rank file of figure 15 at the address 29 indicated in the head address field of the thesaurus entry (figure 14A), and then at the chained addresses 27 and 15. In parallel, the corresponding layer 1 bitmap segments 001, 010 and 100 are read. Figures 15 and 16 also show that the rank and bitmap segment files have an additional chain consisting of free records (addresses 32/33/17 in figure 15 and 29/8/17/24 in figure 16). The head of the latter chain is allocated to write new coding data when necessary.

Preferably, the thesaurus entry further has a layer 1 tail address field for containing the address in the rank file of the last record of the chain pertaining to the entry, as shown in the third columns of figures 14A, 14G and 14H. This facilitates the updating of the encoding data storage. For instance, the insertion of a new layer 1 rank for Max, with a corresponding layer 1 bitmap segment, may proceed as follows: the head of the free record chain is located (address 32); the address (33) found in its next record address field becomes the address of the new free record chain head; the records at address 32 receives the new layer 1 rank in the rank field, the end-of-chain flag (0) in the next address field and the new bitmap segment in the segment field, respectively; the address obtained in the tail address field of Max's thesaurus entry (15) is accessed directly (bypassing the potentially long path along the chain) to write the address (32) of the new data, which is also written into the tail address field of Max's thesaurus entry. The fact that the layer 1 rank is a new one for Max can be determined from the layer 2 data: if the layer 2 updating performed previously has changed a "0" to a "1" in the layer 2 bitmap segment, then the layer 1 rank is a new one for the word; otherwise the layer 1 rank is already present in Max's layer 1 rank list which has to be scanned until said layer 1 rank is found. If there are more than two encoding layers, it is possible to provide a layer k tail address field in the thesaurus entries for $k > 1$ and to proceed in the same manner for new layer k ranks as determined from the layer k+1 data. However the main gain in doing so lies in layer 1 which has the longest chains.

In figures 15 and 16, the coding data coming from three heterogeneous thesauruses (client name thesaurus, accident amount word thesaurus and

WO 02/44943

PCT/IB01/02792

- 28 -

accident amount macroword thesaurus) are stored in the same data containers. The other thesauruses are ignored for clarity of the figures. In fact, all the coding data of one layer may be piled up in the same rank / bitmap segment files, irrespective of the word or macroword thesaurus where they come from.

5 Any entry of any thesaurus will then point to a respective record chain in those two coupled files.

In order to optimize the processing speed, it is preferable to sort the rank and bitmap segment files for disc storage, so as to group the records based on the thesaurus entries to which they pertain. The advantage in doing
10 so is that the reading of the coding data for one thesaurus entry requires fewer disc accesses, by means of the computer cache memory which enables the simultaneous RAM loading of a group of physically contiguous records. A batch execution of that optimization sorting, which requires a simultaneous update of the thesaurus entries (head and tail address fields), may be used to avoid
15 untimely resource usage.

In order to facilitate this optimization, it is preferable to use separate data containers for different thesauruses, rather than common files. This reduces the amount of data to be sorted each time. In particular, using one rank / bitmap segment file pair for each thesaurus and each coding layer
20 seems appropriate.

A further possibility is to provide separate rank and bitmap segment files for the different thesaurus entries. This requires a higher number of file declarations in the memory. But it is optimal in terms of processing speed without requiring the above-mentioned optimization sorting operation. It also
25 eliminates the need for storing head and tail addresses pointing to record chains: the thesaurus entries simply designate data containers where the rank and bitmap segment data are stored.

Figure 17 illustrates how the data of the client name thesaurus may be arranged in the latter case. The thesaurus has an index register where the
30 thesaurus words are kept sorted. For each word and each coding layer k, two files are provided in the system memory, one for containing the rank data (noted NOK), and one for containing the bitmap segments (noted HPk). The attribute value (André, Ariane and so on) can be used to name the

WO 02/44943

PCT/IB01/02792

- 29 -

corresponding files. The storage is less compact than with common data containers as shown in figures 15-16, but access to the data relating to one word can be very quick without requiring any sorting.

5 An arrangement as illustrated in figure 17 is preferred if the operating system does not suffer too severe limitations regarding the number of files that can be managed in the memory, and if the overhead due to the storage of numerous individual files is not a problem. Otherwise, it is possible to group the rank and bitmap segment files relating to different (macro)words, or even to different thesauruses, as indicated before.

10 In addition to enhanced data compression, the multi-layer row-ID list encoding method provides a substantial acceleration of most query processing. The processing is first performed in the higher layer, and the results are passed to the lower layers. The coding scheme preserves a common structure for the entries of all thesauruses in each layer, imprinted by the original structure
15 imparted by the virtual flat file. Accordingly, collective logical operations between integer lists or bitmaps originating from different thesauruses are possible in the various layers. The results obtained in a layer k+1 provide a sort of filter for executing the minimum number of operations in layer k, which enhances the processing efficiency, particularly for multi-attribute query criteria.

20 This enhancement is hardly visible on our simplified example, which is too small. Consider the following request: find Max's accidents for an amount of 1,300 (there is no response). The direct layer 1 processing is to read and decode the relevant layer 1 data to rebuild the bitmap vectors of the words "Max" and "1,300" in the thesauruses of figures 10A and 10G, and to compute
25 the logical AND of the two bitmap vectors. Exactly the same kind of processing in layer 2 requires fewer read operations since there are fewer layer 2 records, and avoids any layer 1 processing because there is no overlap between the two layer 1 rank lists for the words "Max" and "1,300" (2nd column of figures 13A and 13G). If the same request is made with the amount value 10,000
30 instead of 1,300, the layer 2 results may reduce the layer 1 processing to loading the two layer 1 bitmap segments corresponding to rank 0 (the other ranks are filtered out) and computing the AND between those segments.

With more representative values of D1 and D2 (e.g. D1 = D2 = 128)

WO 02/44943

PCT/IB01/02792

- 30 -

and a large size database, this filtering principle between two layers provides a spectacular gain. Large pieces of bitmap vectors disappear from the layer 1 (or generally layer $k \geq 1$) processing owing to the groupwise filtering achieved in layer 2 (layer $k+1$).

5 VDG CREATION AND MANAGEMENT

Figure 18 shows an exemplary layout of a computer system suitable for forming the hardware platform of a system in accordance with the invention. That hardware platform may be of conventional type. It has a bus 100 for exchanging digital signals between a plurality of units including:

- 10 - a central processing unit (CPU) 101;
- a read only memory (ROM) 102 for containing basic operating instructions of the CPU;
- a random access memory (RAM) 103 which provides a working space for the CPU 101, dynamically containing program instructions and variables handled by the CPU;
- 15 - a man-machine interface 104 which comprises circuitry for controlling one or more display devices (or other kind of devices for delivering information to humans) and circuitry for inputting information to the computer system from acquisition devices such as a keyboard, mouse, digital pen, tactile screen, audio interface, etc.;
- 20 - a mass storage device for storing data and computer programs to be loaded into RAM 103. In the typical example shown in figure 18, the mass storage device comprises a hard drive 105 for storing data on a set of magnetic discs 106. It will be appreciated that any kind of mass storage device, magnetic or optical, may be used in implementing the invention.
- 25

For implementing the present invention, the hard drive unit 105 is used for storing data structures as described in the foregoing and programs described in more detail herebelow. The program instructions and the useful data are loaded into the dynamic storage RAM 103 for processing by CPU 101. The query results are stored in the hard drive and/or delivered to a user through the man-machine interface 104 or through a network interface (not

30

WO 02/44943

PCT/IB01/02792

- 31 -

shown) in the case of a remote access.

The mass storage device 105 is suitable for the storage of large amounts of data, but with an access time significantly longer than the RAM 103. This is due to the time needed to put the reading head of the hard drive in front of the desired disc location. As well-known in the art, when a disc access is performed in hard drive 105, the data that are actually read form a block of data stored contiguously on the hard disc, which is loaded in a portion of RAM 103, called "cache" memory. When it is known that the CPU is likely to need different data pieces simultaneously or in a short period of time, it is convenient to arrange the data storage organization such that those data belong to the same block so as to be retrievable by a single disc access, which minimizes the processing time.

The system of figure 18 may be a personal computer (PC) of the desktop or laptop type. It may also be a workstation or a mainframe computer.

Of course, other hardware platforms may be used for implementing the invention. In particular, those skilled in the art will appreciate that many calculations performed on the bitmap segments and vectors lend themselves to efficient implementation by means of dedicated logical circuits or coprocessors. Furthermore, parallel computation is very natural in this system.

The process of creating the VDG data structure is now described with reference to figure 19 from input data tables being in the form shown in figures 1-3, which is the most usual data representation. That creation process is thus suitable for creating the VDG structure from legacy databases. From the VDG updating rules described further on, it will be understood that VDG's may also be created directly from brand new data.

In certain databases, the data tables have their rows characterized by compound keys rather than row-ID's as in figures 1-3. A compound key is the concatenation of the contents of several key fields of a data table. In a source data table, the records include foreign keys which designate the compound keys of records of a target table. If such a legacy databases is handled, the first stage of the VDG creation procedure is to translate the compound keys into single keys such as the row-ID's shown in figures 1-3. This (optional) first stage is illustrated in box 120 in figure 19.

The second stage 121 consists in completing the data tables with *Null* records where appropriate. This is performed as discussed hereabove with reference to figures 4-7, by scanning every path in the data table tree from the leaf table of the path to the root table. A link to a *Null* record is denoted by the default value -1. As a result, for each source / target table pair, all the row-IDs of the target table are present at least once in the source table link column.

The next stage 122 comprises the creation of the word thesauruses. The relevant attributes, i.e. those likely to be used in query criteria (it may be all of them), are determined. For each of the determined attribute, the word format (type and length) is selected. For each word thesaurus, the attribute values occurring in the associated column, including the *Null* value, are read from the data table stored in the hard drive 105. Repeated values are eliminated, and the remaining values are sorted based on the attribute values and the order relationship applicable to the type of attribute. This sorting operation may be performed in successive data record blocks transferred from the hard drive 105 to the CPU cache memory, with an external sorting after processing each block.

The VDG creation procedure then proceeds to a stage 123 of deciding the relevant macroword formats. Some word thesauruses will not give rise to macroword thesauruses (for example, the client gender thesaurus of figure 10C needs no macrowords). In contrast, other thesauruses, for example corresponding to date or amount attributes, will give rise to several macroword thesauruses having different truncation lengths. If the values found in an attribute column include characters strings beginning by most letters of the alphabet, it is convenient to create a macroword thesaurus with a prefix length of one character. The decision about the suitable macroword hierarchy may be made by a database manager and input through the man-machine interface 104. It may also be an automatic process, based on the attribute type and/or the distribution of the words in the thesaurus. In stage 123, the macroword thesauruses are also created, directly in sorted form, by applying the truncation to the words of the corresponding word thesauruses and deleting the repeated macrowords.

Each entry of a macroword thesaurus preferably indicates the first word

WO 02/44943

PCT/IB01/02792

- 33 -

(or lower level macroword) of the lower level thesaurus included in the range covered by the macroword. This indication of the lowest word (or macroword) whose prefix matches the macroword under consideration reduces the time needed to access the "children" of that macroword since the first one can be
 5 accessed without scanning the lower level thesaurus. Alternatively, or cumulatively, the highest word (or lower level macroword) whose prefix matches the macroword could be indicated in the macroword thesaurus.

In stage 124, the rows of the link table and the entries of the individual word thesauruses are generated. This is preferably done without storing the
 10 whole flat file (figure 8), for example according to the algorithm illustrated in figure 20, in the case of an encoding with $n = 2$ layers.

In the embodiments illustrated in figures 20-32, it is assumed that each entry of a thesaurus for an attribute value contains an index WI which forms a row-ID in an auxiliary table of the type shown in figure 14A, 14G or 14H,
 15 pointing to coding data containers of the type shown in figures 15 and 16. For each encoding layer k , this auxiliary table has :

- a column for containing the address, noted $AT\_Fk(WI)$, of a first record concerning the thesaurus word of index WI in the coding data container relating to layer k ;
- 20 - a column for containing the address, noted $AT\_Lk(WI)$, of the last record of the chain for thesaurus word of index WI in the data container; as indicated before, the latter column may be present only for layer 1.

As mentioned previously, the data container for a given coding layer may be shared between all or part of the thesauruses, or it may be associated
 25 with each individual thesaurus. A record at address AD (≥ 1) in the layer k container (here assumed to be common to all thesauruses) comprises a first field $NOK(AD)$ for containing the rank data as an integer ranging from 0 to $Dk-1$, a second field for containing the address $NXk(AD)$ of the next record of the chain (this address is 0 if there is no further address), and a third field for
 30 containing the corresponding bitmap segment $HPk(AD)$. The layer k container has a free record chain whose first record address is noted Hk .

It is noted that the auxiliary table could also be shared by several thesauruses containing distinct word indexes to access such common auxiliary

WO 02/44943

PCT/IB01/02792

- 34 -

table.

Before stage 124, all the records of the data container are chained together and free, and the bitmap segments HPk(AD) are initialized with all zero segments. The columns AT_Fk and AT_Lk of all the auxiliary tables are also initialized with the value 0.

The quotient and the remainder of the Euclidean division of a flat file row-ID by D1 are respectively noted Q1 and R1. For each further layer $k > 1$, Qk and Rk respectively denote the quotient and remainder of the Euclidean division of Q(k-1) by Dk. At the initialization step 130 of figure 20, the integers Q1, R1, Q2 and R2 are set to 0.

The rows of the root table(s), which may be read one by one or block by block from the hard drive 105, are selected one by one in step 131. The records of the other data tables which are linked with the selected root table row are read in step 132. This provides a data graph of the type illustrated in compact form in figures 5-7.

The links of those data graphs, i.e. the row-ID's in the data tables, are written into the relevant columns of the link table (figure 9) at row-ID $Q1 \times D1 + R1$ (step 133). If there is no link table, step 133 is skipped.

For the current data graph, the different attributes AT are successively selected (step 134). The value of the selected attribute AT is located by means of a dichotomic search in the corresponding thesaurus, and its word index WI is read in step 135. Step 136, which will be detailed hereafter with reference to figures 21-24, consists in updating the auxiliary table and data containers with respect to the AT thesaurus entry for the word index WI. This updating corresponds to the insertion of the current flat file row-ID $Q1 \times D1 + R1$ into the integer list relating to the thesaurus word index WI.

When all the attributes have been thus handled (test 137), the layer 1 remainder index R1 is incremented by one unit in step 138. If the incremented R1 is equal to D1 (test 139), then the index R1 is reset to 0, and the layer 1 quotient index Q1 and layer 2 remainder index R2 are each incremented by one unit in step 140. If the incremented R2 is equal to D2 (test 141), then the index R2 is reset to 0, and the layer 2 quotient Q2 is incremented by one unit in step 142. After step 142, or when $R1 < D1$ in step 139 or $R2 < D2$ in step 141,

WO 02/44943

PCT/IB01/02792

- 35 -

a test 143 is performed to determine whether all the rows or all the root tables have been considered. If not, the procedure comes back to step 131 to select a new root table row.

Once all the root table rows have been considered, stage 124 of figure 19 is over, and the parameters $Q1$, $R1$, $Q2$ and $R2$ are memorized for subsequent insertion of possible new data records. Eventually, the number of rows in the virtual flat file is given by $Q1 \times D1 + R1$.

Clearly, the procedure of figure 20 is readily extended to $n > 2$ encoding layers, by initializing all Q_k and R_k parameters to 0 in step 130 and by developing steps 138-142 (which are equivalent to incrementing the data graph pointer $Q1 \times D1 + R1$) in the higher layers.

Figure 21 shows how the program can manage the record chains in the data container and the thesaurus auxiliary table in layer $k \geq 1$ for a word index WI in the thesaurus relating to an attribute AT . The first step 150 is to load the value $AT\_Fk(WI)$ stored in the auxiliary table into the address variable AD . If $AD = 0$ (test 151), then a record chain has to be initialized for thesaurus index WI , so that the head address Hk of the free record chain in the data container is assigned to $AT\_Fk(WI)$ in step 152.

If there was already a record chain for the thesaurus index WI ($AD > 0$ at test 151), the rank $NOk(AD)$ is loaded into the rank variable q in step 153. If the following test 154 shows that q is different from the quotient variable Qk , the address variable AD' receives the address of the next record of the chain, i.e. $NXk(AD)$, in step 155. If AD' is still different from 0 (test 156), the process comes back to step 153 for examining the next rank variable of the record chain, after substituting AD' for AD in step 157. When $AD = 0$ in test 156, a data container record has to be appended to the chain for thesaurus index WI , so that the head address Hk of the free record chain, is written into the next record field $NXk(AD)$ in step 158.

After step 152 or 158, the head address Hk of the free record chain is loaded into the address variable AD in step 159. Step 160 is then executed to update the auxiliary table and data container. This update operation 160 consists in:

WO 02/44943

PCT/IB01/02792

- 36 -

- replacing the head address Hk by the next address NXk(AD) of the free chain;
- writing the current value of the address variable AD into AT_Lk(WI); and
- writing Qk and 0, respectively, in the fields NOk(AD) and NXk(AD) of the data container.

After step 160, or when $q = Q_k$ in the above-mentioned test 154, the bitmap segment HPk(AD) is updated in step 161 by writing the digit "1" at bit position Rk of that segment.

In figure 20, it has been considered that both the layer 1 and layer 2 coding data are updated in step 136. This means that the procedure of figure 20 is executed once for $k = 1$ and once for $k = 2$. Another possibility is to execute it only for $k = 1$, and to generate the layer 2 coding data subsequently, by processing the layer 1 rank data produced in stage 124.

It is worth noting that when initializing the VDG's from a legacy database as in figure 20, the rank data Qk appear in an increasing order (we always have $q \leq Q_k$ in test 154 of figure 21). Accordingly, it is possible to move directly to the record chain tail, i.e. to take $AD = AT\_Lk(WI)$ instead of $AD = AT\_Fk(WI)$ in step 150. In this case, step 158 is executed directly when $Q_k > q$ in test 154, thereby avoiding the scanning of the record chain. Such embodiment is illustrated in figure 22.

In the latter embodiment, once the VDG initialization is over, the layer k tail address fields AT_Lk with $k > 1$ may be discarded. However, if the VDG management is such that any new VDG likely to be inserted has a flat file row-ID equal to or greater than all the flat file row-ID's of the existing VDG's (i.e. the flat file row of any deleted VDG will not be used any more), then it is advantageous to keep all the tail address fields AT_Lk in order to perform any subsequent update in accordance with the embodiment of figure 22.

In the form depicted in figure 21, the update procedure is applicable independently of any hypothesis on the rank values Qk.

Figures 23 and 24 show an alternative method of updating the auxiliary table and data containers with respect to the AT thesaurus entry for the word index WI in step 136, which takes advantage of the tail address field AT_L1 of the auxiliary table in layer 1 (with $n = 2$ coding layers). Figure 23 illustrates the

WO 02/44943

PCT/IB01/02792

- 37 -

layer 2 processing which is performed before the layer 1 processing of figure 24. Most of the steps of figures 23-24 are very similar to steps of figure 21, so that corresponding reference numerals have been used.

- The layer 2 processing of figure 23 is essentially the same as that of figure 21 ($k = 2$), with the following differences:
- it is not necessary to deal with tail address fields $AT\_L2(WI)$ in step 160;
 - step 161 further includes setting to "1" the binary variable $LL1$, which means that the current layer 1 rank data $Q1$ does not belong to the layer 1 record chain relating to the word index WI ;
 - when $q = Q2$ in test 154, another test 164 is made to determine whether the bit position $R2$ of the layer 2 segment $HP2(AD)$ contains the value "1"; step 161 follows only if that test 164 is negative;
 - if test 164 is positive, the current layer 1 rank data $Q1$ already belongs to the layer 1 record chain relating to the word index WI , so that the variable $LL1$ is set to "0" in step 165.

The layer 1 processing of figure 24 begins at step 170 by testing whether $LL1$ is 0 or 1. If $LL1 = 0$, step 150 is executed to load the value $AT\_F1(WI)$ stored in the layer 1 auxiliary table into the address variable AD , and a loop 153-155 is executed to find the data container address AD where the data relating to the rank $Q1$ are stored. Steps 153 and 154 are the same as in figure 21, and in step 155 the next address $NX1(AD)$ is directly loaded into the address variable AD (AD is never 0 because $LL1 = 0$). The program proceeds to step 161 when $q = Q1$ in test 154.

If $LL1 = 1$ in test 170, step 171 is executed to load the value $AT\_L1(WI)$ stored in the layer 1 auxiliary table into the address variable AD . If $AD = 0$ (test 172), the sequence of steps 152, 159-161 is executed as in figure 21 (however, it is not necessary to deal with next address fields $NX1(AD)$ in step 160). If $AD = 1$ in test 172, the sequence of steps 158-161 is executed as in figure 21.

The procedure of figures 23-24 avoids the scanning of the layer 1 record chains when the rank data $Q1$ are not in such chains, without any hypothesis on the rank values.

After all the coding data for the individual word thesauruses have been generated, the next stage 125 of the procedure shown in figure 19 is to

WO 02/44943

PCT/IB01/02792

- 38 -

rearrange the stored coding data. As indicated previously, this is done to organize the record chains in the coding data container of each layer so that records pertaining to the same thesaurus word have contiguous addresses in order to be accessible in one or few disc accesses by means of the CPU cache memory. A simple way to do this is to reserve memory space for a new auxiliary table and new coding data containers. The thesaurus words are considered one by one, and for each of them, the coding data pointed to in the old auxiliary table are read sequentially and copied into the new data container at an address AD incremented after each write operation. When proceeding to the next thesaurus word index $WI+1$, new pointers $AT\_Lk(WI) = AD-1$ and $AT\_Fk(WI+1) = AD$ are determined and stored into the new auxiliary table. After all the coding data records have been thus read and rewritten into the new data container, the old data container and auxiliary table are discarded.

Such rearrangement can be performed separately for each coding layer k.

If there are several data containers for different thesauruses in a coding layer, they may also be reordered separately.

As indicated before, the rearrangement step 125 is dispensed with when the thesauruses are organized in the manner illustrated by figure 17, since the coding data files naturally fulfil the grouping condition with respect to the thesaurus words.

In the following stage 126 of the procedure shown in figure 19, the macroword thesaurus entries are generated. For each macroword and each layer, this is done simply by merging the rank coding data Q1, Q2 of the words (or lower level macrowords) covered by the macroword, and by obtaining the corresponding bitmap segments by a logical OR of those relating to the words (or lower level macrowords). If the coding data have been rearranged for the word thesauruses as indicated in stage 125, the same grouping of the coding data will automatically be achieved for the macroword thesauruses.

In stage 127, the now useless link columns of the original data tables (figures 1-3) can be deleted. The *Null* records which have been added in stage 121 can also be deleted, their occurrence being indicated by the default value -1 in the link table (figure 9).

WO 02/44943

PCT/IB01/02792

- 39 -

Finally, the elements to be stored in the hard drive 105 in the above-described embodiment are:

- the data tables as illustrated in figures 1-3, without the link columns.
Parameters defining the data table tree structure of figure 4 are stored in
5 association with the tables;
- the link table as illustrated in figure 9;
- the sorted thesauruses comprising an index register and an auxiliary
table for each desired attribute. Figures 25-26 show the index registers
for the attributes AT = CN ("client name") and AT = AA ("accident
10 amount") in our simplified example. Figures 28-29 show the
corresponding auxiliary tables;
- the macroword thesauruses organized like the individual word
thesauruses, with a specified truncation or prefix length. The index
register of each macroword thesaurus further has an additional column
15 containing, for each macroword, the row-ID, in the index register of the
thesaurus of lower level for the same attribute, of the first word (or
macroword) covered by the macroword. Figures 27 and 30 show the
index register and auxiliary table for the attribute AT = CN and the
truncation length 3;
- 20 - the coding data container(s) for each coding layer, each having a
variable head address for its free record chain. Figures 31 and 32 show
layer 1 and layer 2 data containers shared by the thesauruses of figures
24-29 (free record chain head addresses 31 and 27, respectively);
- optionally, one or more thesauruses stored in a "low density" format
25 suitable for attributes of high cardinality. In the low density format, $n = 0$
and the flat file row-ID's are stored as explicit (short) integer lists, for
example by means of record chains. If the coding data for layers 1
through n are needed, they are easily calculated by performing n
successive Euclidean divisions from each stored integer of the list. For a
30 given high cardinality attribute, it may be appropriate to provide an
individual word thesaurus in the low density format and one or more
macroword thesauruses in the "normal" encoded format.

The data containers of figures 31 and 32 are derived from those shown

WO 02/44943

PCT/IB01/02792

- 40 -

in figures 15 and 16 pursuant to the rearrangement stage 125, in which the auxiliary tables of figures 28-30 are also obtained from those of figures 14A, 14G and 14H, respectively. For treating a query concerning the client called André, the processor would have to read records #20, #11 and #2 of figure 15 (limiting ourselves to layer 1) if the coding data container were not rearranged, whereas it reads the physically contiguous records #1, #2 and #3 of the rearranged container of figure 31. The latter reading can typically be done by loading a single block into the cache memory. More disc accesses, and hence a longer processing time, are required for reading scattered records.

The columns separated by broken lines in figures 9 and 28-32 are preferably stored separately. For example, the storage address of one value in such a column may be defined as a start address assigned to the column plus an offset equal to its row-ID multiplied by a field length.

Accordingly, the links of a link table row (data graph) are stored at corresponding addresses given by the flat file row-ID. This separate storage of the link column accelerates the data retrieval when some of the data tables need not be accessed to produce the output data requested in the query.

Likewise, some elementary operations performed in the query processing require only coding data for one layer, so that it is advantageous to separate the information concerning each layer in the auxiliary tables to accelerate the processing. Other operations imply the rank data and the bitmap segment data independently, so that it is advantageous to separate those data in the data containers as shown in figures 31 and 32.

In an alternative way of storing a thesaurus, the word index register and the auxiliary table are merged in a single table with a Huffman type of indexing: each row of that table contains a value of attribute AT, the AT_Fk and AT_Lk data, a next row pointer (the next row contains the next value of the attribute in the sorted thesaurus) and optionally a preceding row pointer.

In an embodiment, the maintenance of VDG's created as described hereabove may involve the following operations:

1/ Record insertion

A new virtual data graph, i.e. a new row in the flat file, is generally

WO 02/44943

PCT/IB01/02792

- 41 -

generated in response to the insertion of a new record in a data table.

However, if the new record has a link to an existing record of another target table such that no other link points to said existing record, then there is no need for a new data graph, but for the update of an existing data graph. For example, if client Oscar subscribes a first policy, e.g. for his car, a new record is added to the policy data table without creating any new VDG: the data graph of figure 7 is simply modified to place the new data in the node corresponding to the policy table. If Oscar then subscribes a second policy, e.g. for his house, a new VDG will be necessary.

To generate the new VDG, all records from the other data tables, related to the new inserted record, including *Null* records, are identified by their respective row-ID's which, if necessary, can be retrieved by queries based on attribute values of those related records.

After appending the new record to the data table, the first thing to do is to initialize any new thesaurus entry which may be necessary if new attribute values occur (all AT_Fk and AT_Lk fields are initialized to 0). The new virtual flat file row and its corresponding thesaurus entries may be generated as in steps 133-142 of figure 20. Any higher level macroword thesaurus is updated accordingly.

2/ Record attribute modification

Changing or adding an attribute value in an already existing data table record has no effect on the link table which does not reflect the table contents but the link structure. Adding is a particular case of changing when the preceding attribute value was *Null*. Likewise, deleting an attribute value from a record is a particular case of changing when the new attribute value is *Null*.

If the new attribute value requires a new thesaurus entry, such entry is initialized (AT_Fk = AT_Lk = 0). The list L of the link table row-ID's corresponding to flat file records comprising the data record to be amended is obtained by placing a suitable query. The latter list L is merged (bitmap OR operation) with the flat file row-ID list L' of the new attribute value, and the coding data of the merged list $L \vee L'$ are assigned to the new attribute value. The complement $\bar{L}$ of list L is also determined (bitmap NOT operation) to be

WO 02/44943

PCT/IB01/02792

- 42 -

intersected (ANDed) with the flat file row-ID list L'' of the preceding attribute value. If the resulting intersection list $\bar{L} \wedge L''$ is not empty, its coding data are assigned to the preceding attribute value. This may transfer to the free record chain of one or more data containers records that previously belonged to the record chain associated with the preceding attribute value. If the intersection list $\bar{L} \wedge L''$ is empty, the preceding attribute value may be deleted from its word thesaurus. The same intersection and update sequence is performed for any higher level macroword thesaurus.

3/ Record link modification

Changing a link in a source data table leads to corresponding changes in every occurrence of the link in the link table. The list L of the concerned link table rows can be determined by processing a suitable query.

If the target table record pointed to by the former link has no more link pointing thereto (its row-ID does not occur any more in the corresponding column of the link table after the modification), a new VDG is generated. Downstream of the modified link, this new VDG has the same content as the one(s) which is (are) being amended. Upstream of the modified link, it consists of *Null* records. The new virtual flat file row and its corresponding thesaurus entries may be generated as in steps 133-142 of figure 20. Any higher level macroword thesaurus is updated accordingly.

After that, a procedure similar to the one described in the preceding section can be performed for each attribute of the target table: /a/ the list L is merged with the flat file row-ID list L' of the new attribute value (the value occurring in the target table record pointed to by the new link); /b/ the coding data of the merged list $L \vee L'$ are assigned to the new attribute value; /c/ the complement $\bar{L}$ of list L is intersected with the flat file row-ID list L'' of the preceding attribute value (the value occurring in the target table record pointed to by the former link); /d/ the coding data of the resulting intersection list $\bar{L} \wedge L''$ are assigned to the preceding attribute value; and /e/ the same intersection and update sequence is performed for any higher level macroword thesaurus.

If the first target table (for the modified link) has a link column to a second target table, the link value stored in the column of the link table

WO 02/44943

PCT/IB01/02792

- 43 -

associated with the second target table and in each row of list L is also changed, and the above procedure /a/-/e/ is performed for each attribute of the second target table. This is repeated for any data table located downstream of the first target table in the data table tree (figure 4).

5 For example, if a correction is made in the accident table of figure 3 to indicate that accident #6 was under policy #2 instead of policy #0, i.e. concerned Max's house instead of Ariane's car, the link from the accident table in the data graph of figure 5 has to be changed to point to policy record #2, and the link from the policy table has to be changed to point to client record #4. A
10 new row is inserted in the virtual flat file, to contain the useful information about Ariane's car policy under which no accident took place. As a result, row #6 of the link table of figure 9 is changed to include the values 4, 2 and 6, respectively, in the client, policy and accident link columns, and a new row #12 is added including the values 2, 4 and -1, respectively, with corresponding
15 changes in the thesauruses.

4/ Record cancellation

Canceling a record from a root table involves deleting the row(s) of the flat file containing that record. The corresponding flat file row-ID's are removed from the lists encoded in the thesauruses, i.e. zeroes are written at the
20 associated locations of the bitmap vectors. These flat file row-ID's may be made available for further VDG insertion, for example pursuant to section 1/ or 3/ hereabove. They may also remain as blank rows if the virtual flat file size is not a major concern. Likewise, canceling a record from a target table which has no link pointing thereto in the corresponding source table involves deleting the
25 row(s) of the flat file containing that record (these row were representing data graphs with *Null* records upstream of the cancelled record).

If the cancelled record belongs to a target table for a compulsory link (e.g. the client or policy table in our example), any flat file row containing that record is also deleted. If the cancelled record belongs to a target table for an
30 optional link (e.g. the third party or broker table in the example of figure 4), the cancellation comprises a link modification whereby any link pointing to that record is replaced by a link to a *Null* record (link value = -1). Such modification

WO 02/44943

PCT/IB01/02792

- 44 -

may be performed as described in the above section 3/ (but without generating any new VDG).

For any link of the cancelled record which pointed to a non-*Null* target table record whose row-ID does not occur any more in the corresponding column of the link table, it is necessary to generate a new VDG containing the same data as the cancelled record in and downstream of said non-*Null* target table record and *Null* values in and upstream of the cancelled record. The new virtual flat file row and its corresponding thesaurus entries may be generated as in steps 133-142 of figure 20. Any higher level macroword thesaurus is updated accordingly.

5/ Thesaurus update and sorting

With the above-described structure of the thesaurus entries, the cancellation of a word in a thesaurus, which occurs when its flat file row-ID list becomes empty, could be done by leaving the thesaurus entry with zeroes in its HPk data. However, this is not optimal regarding memory usage.

A more efficient method is to update the record chains in the data container, so that the auxiliary table has $AT\_Fk(WI) = AT\_Lk(WI) = 0$ for the entry WI of the cancelled word. In such a case, the word index WI can be released, a default value (e.g. -1) being written into the word index column for the cancelled word in the thesaurus index register.

The creation of a new word thesaurus entry can be done as illustrated in figures 21-24 (AD = 0 in test 151 or 172). The word index WI is obtained by incrementing a counter representing the number of thesaurus entries, or by selecting an available word index (e.g. which has been released previously when canceling another word). In this process, a (useful) row is added to the auxiliary table of the corresponding attribute, with row-ID = WI.

Similar procedures can be applied for updating the macroword thesauruses. A macroword index WI may be released when canceling a macroword (all its constituent words have been cancelled). In the case of a word creation, it is first checked whether the macroword already exists, in which case its macroword index WI is recovered; otherwise, a macroword is also created.

WO 02/44943

PCT/IB01/02792

- 45 -

It is thus appreciated that, once words have been removed and/or added, the auxiliary tables are no more sorted in the ascending order of the thesaurus words. The word index register has to be manipulated in order to maintain the thesaurus sorting.

5 However, it is not necessary to perform such manipulation of the word index register immediately. This is very advantageous because the updated database is made available for any new query without requiring a sorting operation in the whole thesaurus, which may take some time.

10 The newly created words or macroword of a thesaurus can have their word indexes stored in a separate, secondary index register, whereas they share the same auxiliary table and coding data containers as the former words of the thesaurus. Only this secondary index register can be sorted when a thesaurus entry is added, which is a relatively light job since most of the thesaurus words belong to the primary register. When a word is deleted, its row
15 in the primary or secondary index register remains with the default value in the word index column. Accordingly, to access the coding data relating to a given word range, the range boundaries are searched, by dichotomy, in both the primary and secondary index registers to determine the relevant word indexes which are then used in the usual way to address the common auxiliary table
20 and data containers.

From time to time, when the CPU 101 is available, a batch task is run to merge the primary and secondary index registers while deleting their rows having the default value in the word index column. This is a straightforward external sorting operation since both registers are already sorted. The resulting
25 merged register is saved to replace the primary register, and the secondary register is cancelled.

If the secondary word index register becomes too big (i.e. its sorting requires a too long time every time an entry is added) before such merge operation is carried out, it is possible to create a further, tertiary index register
30 to receive the new thesaurus entries, and so forth.

6/ Data container optimization

This is useful if the thesaurus organization is of the type shown in

WO 02/44943

PCT/IB01/02792

- 46 -

figures 25-32 rather than of the type shown in figure 17.

As records are inserted and deleted in a coding data container, the above-mentioned condition that the record chains should preferably be arranged so that records pertaining to the same thesaurus word have
 5 contiguous addresses is no more fulfilled. This does not prevent the database system from operating satisfactorily. However, in order to optimize the query processing time, it is preferable to rearrange the records of the coding data container and the corresponding columns of the thesaurus auxiliary table(s) as in the above-described step 125. Like the word index register sorting, such
 10 rearrangement can be carried out when CPU time is available.

ALTERNATIVE THESAURUS ARRANGEMENTS

If the thesauruses are arranged according to the preferred organization illustrated by figure 17, with distinct files for each word or macroword, the flow charts of figures 19-24 are somewhat simplified. First, stage 125 of figure 19 is
 15 not performed (it is an advantage of the file organization to dispense with such sorting when the VDG's are created and maintained). In figure 20, the dichotomy search 135 and the thesaurus update of step 136 may be replaced by the procedure illustrated in figure 33.

In this procedure, $\text{imax}(\text{AT}, \text{W}, \text{k})$ designates the current number of
 20 layer k records in the coding data file relating to thesaurus AT and word W. These parameters are set to zero for all values of AT, W and k at the initialization step 130.

The value in the current data graph of the attribute AT selected in step 134 of figure 20 is allocated to the variable W in step 175 of figure 33, and the
 25 coding layer index k is initialized to 1. The integer i, which points to the records of the coding data file is first set to zero in step 176. If $i = \text{imax}(\text{AT}, \text{W}, \text{k})$ in the following test 177, a record $\text{AT\_W\_NOK}(i)$ having the value Qk is appended to the layer k rank file pertaining to word W and a record $\text{AT\_W\_HPK}(i)$ having the all-zero value is appended to the corresponding bitmap segment file. This is
 30 done in step 178, where $\text{imax}(\text{AT}, \text{W}, \text{k})$ is also incremented by one unit. If $i < \text{imax}(\text{AT}, \text{W}, \text{k})$ in test 177, the rank $\text{AT\_W\_NOK}(i)$ is loaded into the rank variable q in step 179. If the following test 180 shows that q is different from the

WO 02/44943

PCT/IB01/02792

- 47 -

quotient variable Q_k , the integer i is incremented by one unit in step 181 and the process comes back to step 177 for examining the next rank variable of the file, if any. Accordingly, the scanning of the coding data record chain for each layer k (corresponding to loop 153-156 in figure 21) is performed within the

5 AT_W_NOK file which is smaller than the data container common to all words of the thesaurus. Therefore, the minimum number of disc accesses is ensured.

After step 178, or when $q = Q_k$ in test 180, a "1" is written into the bit of rank R_k of the bitmap segment AT_W_HPK(i) in the relevant coding data file (step 182). The coding layer index k is compared with n (or to a lower value if

10 the higher layer coding data are calculated afterwards) in test 183. If $k < n$, the index k is incremented by one unit in step 184 before coming back to step 176. When $k = n$, the thesaurus update is over and the program proceeds to step 137 of figure 20.

In the procedure of figure 33, the rank data AT_W_NOK(i), each

15 consisting of an integer value, can be read in large blocks from the hard drive 105 to the cache memory, so that the procedure is very quick.

Another option which can be used in the thesauruses is to include in each entry relating to a word an indication of the representation format of the flat file row-ID list. Indeed, the format (e.g. low or normal density) can be

20 chosen word by word depending on the number of data graphs including the word under consideration. This is illustrated in broken lines in the right part of figures 25-27 in the case where there are only two formats, i.e. low density (0) and normal density with $n = 2$ coding layers (1). In the example, all the thesaurus entries are in the normal density format. There could be more than

25 two formats; for example, the format data in the thesaurus could specify the number of coding layers for each word. When the flat file row-ID list are represented by data stored in data containers common to one or more thesauruses, distinct containers are provided for the different coding formats.

When the above option is used, the format for each thesaurus entry

30 can be modified as the database lives, in order to optimize the storage. This is a low priority task since the query engine can work with any format. For example, when thesaurus entries are being updated, it is possible to mark any

WO 02/44943

PCT/IB01/02792

- 48 -

entry for which a format change appears to be desirable, based on predefined conditions fulfilled by the density of the word in the amended database. For example, a word or macroword could be changed from low to normal density format when a certain number of data graphs are identified in its thesaurus entry, and vice versa. Afterwards, when processor time is available, the marked entries can be translated into the new format to optimize the system.

It has been mentioned above that, when $n > 1$, storing the rank data in every coding layer is somewhat redundant, since the flat file row-ID lists are completely defined by the bitmap segment data in all layers and the rank data in the last layer.

Figures 34A-B illustrate an alternative way of arranging the coding data files, which avoids storing the layer k ranks with $k < n$. In this arrangement, it is sufficient that the auxiliary tables (figures 28-30) point to a first record in the layer n data container: the addresses AT_F1 and AT_L1 are not necessary. The data container of the highest layer $n = 2$, shown in figure 34A, is the same as that of figure 32, with an additional field in each record to contain the head address $F(n-1)(AD) = F1(AD)$ of a record chain in the data container of the lower layer $n-1 = 1$. The latter data container (figure 34B) has one record chain for each layer n rank pertaining to each thesaurus entry covered by the data container. Each record of a layer $k < n$ data container comprises a first field for containing the address $NXk(AD)$ of the next record of the chain (this address is 0 if there is no further address), and a second field for containing the corresponding bitmap segment $HPk(AD)$. The layer $k < n$ chain is ordered in accordance with the non-zero bits of the bitmap segment $HP(k+1)$ stored in the record of the upper layer data container which contains the head address of the chain. If $1 < k < n$ (not shown), the record further has a third field for containing the head address of a record chain in the data container of the lower layer $k-1$ (and so forth until $k = 1$).

The procedure for retrieving a flat file row-ID list from a thesaurus pointing to data containers of figures 34A-B may be as follows. The word index WI is used to obtain the address of the first relevant record in the layer 2 data container. For this address (and then for each address of the chain defined by

WO 02/44943

PCT/IB01/02792

- 49 -

the NX2 field), the layer 2 rank NO2 is read and the bitmap segment HP2 is scanned. Every time a "1" is found in this scanning, at a bit position R2, a layer 1 rank $NO1 = NO2 \times D2 + R2$ is determined and a corresponding record of the lower layer data container is read (the first time at the head address given by the column F1 in the layer 2 data container, and then at the addresses pointed to by the NX1 addresses in the layer 1 data container). By this method the layer 1 bitmap segments HP1 and their positions NO1 are retrieved to assemble the bitmap vector representing the desired flat file row-ID list.

In the general case, the data containers are accessed from layer n.
 Each segment HPk read after determining a rank NOk with $k > 1$ is scanned to locate its non-zero bits. Each non-zero bit of HPk located in a position Rk provides a lower layer rank $NO(k-1) = NOk \times Dk + Rk$, and a corresponding bitmap segment HP(k-1) is read in the chain designated in the lower layer container. The process is repeated recursively until $k = 1$: the numbers $NO1 \times D1 + R1$ are the flat file row-ID's for the thesaurus entry.

The coding data files illustrated in figures 34A-B can be created by a method similar to that described with reference to figures 19-21. All the HPk and F(k-1) fields are initialized with zeroes before stage 124. The procedure of figure 21 is executed only for $k = n$, with step 161 replaced by the loop depicted in figure 35 in which the coding layer index k decreases from n to 1.

The first step 450 of this loop consists in writing the digit "1" at bit position Rk of the bitmap segment HPk(AD). If the coding layer k is greater than 1 (test 451), it is decremented by one unit in step 452, and the first address $M = Fk(AD)$ is read in the layer (k+1) coding data container (step 453).

If M is zero (test 454), the head address Hk of the free record chain in the layer k coding data container is written into the first address field Fk(AD) of the layer (k+1) coding data container (step 455), to create a new chain. The value of AD is then replaced by Hk (step 456), and the record chains are updated in the layer k coding data container (steps 457-458): Hk is replaced by NXk(AD) before NXk(AD) is set to 0. After step 458, the process loops back to step 450.

If $M > 0$ in test 454, the index R is set to 0 in step 460 to initialize the

WO 02/44943

PCT/IB01/02792

- 50 -

scanning of the bitmap segment $HP = HP(k+1)(AD)$. If R is smaller than the remainder $R(k+1)$ corresponding to the current data graph identifier, the corresponding bit $HP(R)$ of the bitmap segment HP is evaluated (test 462). If $HP(R) = 0$, the program proceeds to step 463 for incrementing R by one unit
 5 before coming back to test 461. When $HP(R) = 1$ in test 462, it is necessary to move forward in the layer k record chain: the integer M' receives the value of M in step 464, and M is replaced by $NXk(M')$ in step 465. If the new value of M is not zero (test 466), the program proceeds to the above-mentioned step 463. Otherwise, the end of the layer k record chain is reached, so that the head
 10 address Hk of the layer k free record chain is assigned to $NXk(M')$ in step 467 before proceeding to the above-mentioned step 456.

If R is equal to the remainder $R(k+1)$ in test 461, the corresponding bit $HP(R)$ of the bitmap segment HP is also evaluated (test 470). If $HP(R) = 1$, the rank Qk already exists in the layer $k+1$ input list relating to the current
 15 thesaurus entry, so that it is not necessary to create a new record in the layer k coding data container: the value of AD is simply replaced by M in step 471, and the process loops back to step 450.

If $HP(R) = 0$ in test 470, the value of AD is replaced by the head address Hk of the free record chain (step 472), and the Huffman-type record
 20 chains are updated in the layer k coding data container (steps 473-474): Hk is replaced by $NXk(AD)$ before $NXk(AD)$ is set to M . After step 474, the process loops back to step 450.

The loop of figure 35 is over when $k = 1$ in test 451.

Figures 34C and 34D show tables whose contents are equivalent to
 25 those of figures 34A and 34B, and in which the bitmap segments HPk for $k > 1$ are not explicitly stored. The layer n coding data container (figure 34C) is identical to that described with reference to figure 34A, but without the HPn column. Each layer k coding data container for $k < n$ (figure 34D) is identical to that described with reference to figure 34B, with an additional column $R(k+1)$
 30 containing layer $k+1$ remainders. The presence of a remainder value $R(k+1)$ in a record to the layer k coding data container means that there is a "1" at position $R(k+1)$ in the non-stored higher layer bitmap segment $HP(k+1)$.

It will be appreciated that the scheme of figure 17, i.e. distinct coding

WO 02/44943

PCT/IB01/02792

- 51 -

data files for each thesaurus entry to minimize the disc accesses, is also applicable when the stored coding data do not include the ranks for layers 1, ..., n-1. The layer n ranks and bitmap segments may be stored as in figure 17. For the lower layers, there are several options. There can be one data container for each thesaurus word and each coding layer $k < n$, with record chains pointed to in the records relating the upper layer $k+1$. The layer k record chains can also be isolated in distinct files whose name include the attribute name AT, the word or macroword value W, the coding layer index k and a layer $k+1$ rank $NO(k+1)$. Each record of such file $AT\_W\_k\_NO(k+1)$ then contains a layer $k+1$ remainder $R(k+1)$ and a layer k bitmap segment HPk which is located at rank $NQk = NO(k+1) \times D(k+1) + R(k+1)$.

QUERY CRITERIA HANDLING

As in any RDBMS, queries can be expressed in accordance with the Structured Query Language (SQL), which has been adopted as a standard by the International Standard Organization (ISO) and the American National Standard Institute (ANSI).

A general flow chart of the query processing procedure is shown in figure 36.

The query criteria, contained in the SQL "WHERE" clause, are converted into a request tree in stage 190 of figure 36. The query criteria are analyzed and structured according to a tree in which the leaves correspond to ranges for respective attributes values as defined in the SQL query and the nodes correspond to logical operations to be performed from those leaves. The leaves are also referred to as "BETWEEN clauses" of the SQL query. An individual attribute value defined in the SQL query is a BETWEEN clause covering a single word.

Example 3:

An example of such a tree is shown in figure 37 in the illustrative case of a query which consists in finding all data graphs relating to accidents undergone by client André or client Max and having a damage amount AA such that $500 \leq AA \leq 5000$. That tree has three leaves, indicated by broken lines,

WO 02/44943

PCT/IB01/02792

- 52 -

corresponding to the BETWEEN clauses defined in the query: [André, André] and [Max, Max] for the client name attribute and [500, 5000] for the accident amount attribute. The tree also has two nodes, one for the OR operation between the two CN criteria, and one at the root for the AND operation with the
5 AA criterion.

The tree decomposition is not unique. The one having the minimum number of nodes is preferably selected.

The next stage 191 is a tree expansion made by analyzing and splitting the BETWEEN clauses relating to attributes having macroword thesauruses.
10 This is done from the tree obtained in step 190, with reference to the sorted thesaurus word and macroword index files associated with the attributes used in the query. The lower and upper bounds of each range defined in a BETWEEN clause are compared with the words of the thesaurus associated with the attribute, to find a decomposition of the range into sub-ranges,
15 whereby each sub-range is also defined as a BETWEEN clause in a word or macroword thesaurus.

In a preferred embodiment, the decomposition is optimized to make maximum use of the macrowords. This optimization consists in retaining the lowest possible number of words or macrowords to form the sub-ranges to be
20 mapped onto the range defined in the BETWEEN clause. The system selects the highest level macrowords that are included in the interval, and repeats the same process in the remaining parts of the range until the atomic word level is attained or the words of the range are exhausted.

In the expanded tree produced in stage 191, the BETWEEN leaves
25 which have been split are replaced by sub-trees made of OR nodes and leaves associated with the sub-ranges. Those leaves are also in the form of BETWEEN clauses, covering thesaurus entries relevant to the query. The expanded tree defines a combination of the relevant thesaurus entries for the subsequent processing.

30 All the leaves of the expanded tree are associated with respective word or macroword (sub-)ranges. Such range may be defined by its bounds in terms of word or macroword row-ID's in the thesaurus index file.

Figure 38 shows the expanded tree corresponding to the tree of Example 3 (figure 37). It is obtained by means of the thesaurus index files of figures 25-27. The one-word ranges "CN = André" and "CN = Max" are not split, but simply encoded by the row-ID's CN_x = 1 and 4 of the words in the thesaurus index file, obtained by dichotomic searches. Another search in the accident amount thesauruses of figures 26 and 27 leads to splitting the range $500 \leq AA \leq 5000$ into three sub-ranges, one for the individual words AA_x = 2 and 5, and one for the macroword AA_3_x = 1.

Figure 39 shows a flow chart of an optimal procedure for splitting a BETWEEN clause in stage 191 of figure 36. It is assumed that the (connected) range does not include the *Null* value (otherwise the leaf can be first split into two substitute leaves linked by an OR node, one leaf with the individual word row-ID AT_x = 0, and the other satisfying the above assumption).

It is also assumed that the attribute AT considered in the BETWEEN clause has a number $Q \geq 0$ of macroword thesauruses indexed by an integer level parameter q with $1 \leq q \leq Q$, the level $q = 0$ designating the individual word thesaurus. For a level q thesaurus, the prefix length (e.g. number of ASCII characters) is noted $P(q)$, with $P(0) > P(1) > \dots > P(Q)$. $P(0)$ is the individual word length. In figures 39-40, x_{max} designates the number of non-*Null* words in thesaurus 0, $W_q(x)$ designates the (macro)word stored at row-ID = x in the level q thesaurus, and $[W]_{P(q)}$ designates the macroword obtained by truncating a word W to keep its prefix of length $P(q)$, for $q \geq 1$.

In the initial step 200 of the procedure of figure 39, the program selects the word thesaurus row-ID's a and b such that $W_0(a)$ and $W_0(b)$ are respectively the lowest and highest thesaurus words included in the range defined for the leaf being processed. The integers a and b are readily selected by dichotomic searches in the word thesaurus based on the range bounds. If the search shows that the range covers no thesaurus word, the procedure is terminated by specifying that the leaf output will be an empty flat file row-ID list.

If $W_0(a)$ is the lowest word of the thesaurus ($a = 1$ in test 201), the binary variable XL is initialized as $XL = 0$ in step 202. Otherwise, it is initialized

WO 02/44943

PCT/IB01/02792

- 54 -

as $XL = 1$ in step 203. If $W_0(b)$ is the highest word of the thesaurus ($b = x_{\max}$ in test 204), the binary variable XR is initialized as $XR = 0$ in step 205. Otherwise, it is initialized as $XR = 1$ in step 206. In the following steps, the value XL (XR) = 0 denotes the fact that the lower (upper) bound of the range under consideration is aligned with a macroword boundary. If it is aligned with a macroword boundary from a level q thesaurus, then this is also true for any level q' thesaurus with $1 \leq q' \leq q$. The initialization 201-206 is valid for $q = Q$.

In step 207, the program invokes a function $FUNC$ whose flow chart is represented in figure 40. This function returns data describing a sub-tree to be inserted in the place of the processed leaf (step 208). The function $FUNC$ has six arguments input when starting its execution in step 210 of figure 40A: the attribute reference AT ; a thesaurus level parameter q ($q = Q$ when the function is first invoked in step 207 of figure 39); the thesaurus row-ID's a and b of the lowest and highest AT words in the range of interest; and the above-defined variables XL and XR .

After step 210, it is determined whether the thesaurus level parameter q is zero (test 211). If $q > 0$, two macroword thesaurus row-ID's a' and b' are selected in step 212, such that $W_q(a') = [W_0(a)]_{P(q)}$ and $W_q(b') = [W_0(b)]_{P(q)}$. This is done by simple dichotomic searches in the level q thesaurus after truncating the words $W_0(a)$ and $W_0(b)$.

In the following test 213, the variable XL is evaluated. If $XL = 1$, it is determined in test 214 whether the consecutive words $W_0(a-1)$ and $W_0(a)$ share the same level q macroword, i.e. whether $[W_0(a-1)]_{P(q)} = W_q(a')$. If so, the integer a' is increased by one unit in step 215. If $[W_0(a-1)]_{P(q)} < W_q(a')$ in test 214, the value of XL is changed to 0 in step 216 since the lower bound of the range under consideration is aligned with a level q macroword boundary. After step 215 or 216, or when $XL = 0$ in test 213, the variable XR is evaluated (test 217). If $XR = 1$, it is determined in test 218 whether the consecutive words $W_0(b)$ and $W_0(b+1)$ share the same level q macroword, i.e. whether $[W_0(b+1)]_{P(q)} = W_q(b')$. If so, the integer b' is decreased by one unit in step 219. If $[W_0(b+1)]_{P(q)} > W_q(b')$ in test 218, the value of XR is changed to 0 in

WO 02/44943

PCT/IB01/02792

- 55 -

step 216 since the upper bound of the range under consideration is aligned with a level q macroword boundary.

After step 219 or 220, or when $XR = 0$ in test 217, the variables a' and b' are compared in test 221. If $a' > b'$, no level q macroword is spanned by the
 5 range under consideration, the program decrements q by one unit in step 222 and comes back to step 211.

When $a' \leq b'$ in test 221, a sub-range of $b' - a' + 1$ macrowords is generated for insertion into the expanded query tree (step 223 in figure 40B). This sub-range covers the macroword row-ID's from $AT\_P(q)\_x = a'$ to
 10 $AT\_P(q)\_x = b'$.

Afterwards, the variable XL is evaluated again in step 224. If $XL = 1$, another range has to be considered, below the sub-range generated in step 223. In step 225, the row-ID b'' of the upper bound of that lower range is determined: the corresponding word $W_0(b'')$ is the highest of the AT thesaurus
 15 such that $[W_0(b'')]_{P(q)} < W_0(a')$. The function $FUNC(AT, q-1, a, b'', 1, 0)$ is then called recursively in step 226, to deal with the additional lower range. After step 226, or when $XL = 0$ in test 224, the variable XR is evaluated again in step 227. If $XR = 1$, another range has to be considered, above the sub-range generated
 20 in step 223. In step 228, the row-ID a'' of the lower bound of that upper range is determined: the corresponding word $W_0(a'')$ is the lowest of the AT thesaurus such that $[W_0(a'')]_{P(q)} > W_0(b')$. The function $FUNC(AT, q-1, a'', b, 0, 1)$ is then called recursively in step 229, to deal with the additional upper range.

When $q = 0$ in test 211, a sub-range of $b - a + 1$ words is generated for insertion into the expanded query tree (step 230). This sub-range covers the
 25 individual word row-ID's from $AT\_x = a$ to $AT\_x = b$.

After step 229 or 230, or when $XR = 0$ in test 227, the execution of the function FUNC is terminated in step 231 by returning the data describing the sub-tree, which have been generated in step 223 or 230 and/or which have been returned by the function recursively called in steps 226 and/or 229.

30 Once the stage 191 of analyzing and expanding the query tree is completed, the expanded tree is processed in stage 192 of figure 36, starting from the highest coding layer n. If $n > 1$, the processing is performed

WO 02/44943

PCT/IB01/02792

- 56 -

successively in the layers k , with k decreasing from n to 1, as shown in the loop represented in figure 41.

The coding layer index k is initialized with the value n in step 240 of figure 41. The layer k processing is started in step 241 by selecting the root ND of the expanded query tree as a node for calling a function named FNODE (step 242). The input to this function comprise the coding layer index k , the parameters describing node ND and its children nodes, and a bitmap vector Res (initialized in an arbitrary manner for $k = n$). Its output is a bitmap vector noted WZ. In layer 1, the bits of value 1 of the output bitmap vector WZ indicate the VDG's (flat file row-ID's) matching the query criteria defined by the tree whose root is node ND. In layer $k > 1$, they indicate the respective layer $k-1$ ranks of the groups of Δk flat file row-ID's which include at least one matching flat file row-ID. In each coding layer index k , the function FNODE is called recursively to process all the nodes of the expanded query tree.

The bitmap vector WZ output by the function called in step 242 is saved as the layer k query result Res in step 243, to be used in the subsequent layer $k-1$ processing if $k > 1$. If so (test 244), the index k is decremented by one unit in step 245, and the next layer processing is started from step 241.

For $k = 1$, Res is the bitmap representation of the desired flat file row-ID list, output in step 246.

A flow chart of function FNODE is shown in figure 42. The bitmap vector WZ is considered there as a succession of segments of D_k bits. The segment of rank N of vector WZ (i.e. the $(N+1)$ -th segment with $N \geq 0$) is noted $WZ[N]$. The bit of rank N of vector WZ (i.e. the $(N+1)$ -th bit with $N \geq 0$) is noted $WZ(N)$. After the function is started (step 248), a working zone is reserved in RAM 103 for containing the bitmap vector WZ (step 249).

In test 250, it is first determined whether ND designates a preset node. A preset node (not illustrated in the example of figure 38) is a node for which a flat file row-ID list has already been determined. Typically, that list has been produced as a matching data graph identifier list in the processing of a previous query (output of step 192). It may also be a combination of such matching identifier lists. One or more preset nodes can be defined in the conversion step

WO 02/44943

PCT/IB01/02792

- 57 -

190 when the SQL query refers to the results of one or more previous queries, for example to restrict the response to records which were included in the response to the previous queries. This feature is particularly useful when the database is used in interactive mode.

5 The flat file row-ID list previously determined for a preset node can be stored in RAM 103 or saved in hard drive 105 (preferably in compressed form in the latter case). That list is encoded according to the n coding layers to provide layer k input lists in the form of bitmap vectors for $1 \leq k \leq n$. Such layer k bitmap vector is loaded as WZ in step 251 when test 250 reveals that the
10 node ND is preset.

 Otherwise, if ND does not designate a leaf but an operator node (test 252), its first child node ND1 is selected in step 253, and the function FNODE is called recursively in step 254 to obtain the bitmap vector WZ1 corresponding to node ND1. The second child node ND2 of the operator node ND is then
15 selected in step 255, and the function FNODE is called again in step 256 to obtain the bitmap vector WZ2 corresponding to node ND2.

 In step 257, the bitmap vectors WZ1 and WZ2 are combined bitwise to form the bitmap vector WZ. The combination $(WZ(N) = WZ1(N) \otimes WZ2(N)$ for any N) is in accordance with the Boolean operator $\otimes$ described in the
20 parameters of node ND, e.g. AND, OR, Exclusive OR, etc. operation. It is essentially a superposition of bitmap vectors, which is performed very quickly since both operand vectors are stored in RAM 103. In step 258, the RAM space which has been allocated to working zones WZ1 and WZ2 is released. In figure 42, the case where the operator node has two child nodes is only considered.
25 Clearly it can be extended to the case where there are more than two operands. Moreover, some operations may involve a single operand, such as the NOT operation, so that the function FNODE may be call only once.

 When node ND is a leaf (test 252), all the bits of the working zone WZ are set to zero in the initialization step 260. In addition, the thesaurus pointer x
30 is initialized to the value $x1$ of the first row-ID of the BETWEEN range defined for node ND.

 If node ND relates to an attribute AT and macroword index q for which

WO 02/44943

PCT/IB01/02792

- 58 -

the thesaurus is stored in the "low density" format (test 261), the leaf processing is as described below with reference to figure 43 (step 262) to obtain the relevant bitmap vector WZ. If the thesaurus format is "normal density", the processing depends on whether the program is in the (chronologically) first layer, that is $k = n$ (test 263). The processing of figure 44 is applied if $k = n$ (step 264), and that of figure 45 if $k < n$ (step 265).

After step 251, 258, 262, 264 or 265, the execution of function FNODE is terminated in step 266 by returning the bitmap vector WZ.

For explaining the low density processing, we assume in figure 43 that the thesaurus storage also makes use of record chains: the thesaurus has an index file similar to those of figures 25-27 (the word index stored at row-ID x being noted $AT\_WI(x)$) and an auxiliary table addressed by the word indexes and containing the addresses $AT\_F(WI)$ in a data container of the first flat file row-ID's of the record chains. In each record of address $AD > 0$, this data container has, in addition to a flat file row-ID value $NO(AD)$, a next address field for containing a pointer to the next address $NX(AD)$ of the record chain. The chain tail has $NX(AD) = 0$. Alternatively, the low density lists could be stored in individual files for each word (similarly to figure 17).

The low density processing of figure 43 has a loop in which the words of the BETWEEN range are successively handled. In each iteration, the program first obtains the word index $WI = AT\_WI(x)$ in step 270, and then the head address $AD = AT\_F(WI)$ in step 271 to initiate the scanning of the record chain. If $AD > 0$ (test 272), there remains at least one item to be examined in the record chain, so that the flat file row-ID value $NO(AD)$ and the next address $NX(AD)$ are read as variables N and M , respectively, in step 273. The

Euclidean division of N by $\Delta k = \prod_{k'=1}^{k-1} Dk'$ ($\Delta 1 = 1$) is made in step 274 to obtain

the layer $k-1$ quotient (rank) N' . For $k = 1$, $N' = N$. For $k > 1$, this operation 274 is simply a deletion of the $\sum_{k'=1}^{k-1} \delta k'$ least significant bits of N (remainder) if the

layer k' divisors Dk' are $2^{\delta k'}$ with $\delta k'$ integer ($1 \leq k' < k$). A "1" is then written into bit $WZ(N')$ of the bitmap vector WZ (step 275). The next address M is

WO 02/44943

PCT/IB01/02792

- 59 -

substituted for AD in step 276 before coming back to the test 272. When the record chain has been completely examined (AD = 0 in test 272), it is determined whether the current word x is the last one x2 of the BETWEEN range (test 277). If $x < x2$, the thesaurus pointer x is incremented by one unit in
 5 step 278 for the next iteration of the loop. The loop is over when $x = x2$ in test 277, and the program proceeds to step 266 of figure 42.

The layer n normal density processing of figure 44 has a similar loop in which the words or macrowords of the BETWEEN range are successively handled, but without recalculating the (stored) coding data. In each iteration,
 10 the program first obtains the word index $WI = AT\_P(q)\_WI(x)$ in step 280, and then the head address $AD = AT\_P(q)\_Fn(WI)$ in step 281 to initiate the scanning of the record chain. If $AD > 0$ (test 282), there remains at least one item to be examined in the record chain, so that the layer n rank value $NOn(AD)$, the next address $NXn(AD)$ and the corresponding layer n bitmap segment $HPn(AD)$ are read as variables N, M and H, respectively, in step 283.
 15 The bitmap segment H is then superimposed, by an Boolean OR operation, onto the segment $WZ[N]$ of bitmap vector WZ (step 284), and M is substituted for AD in step 285 before coming back to test 282. When the record chain has been completely examined (AD = 0 in test 282), it is determined whether the
 20 current word x is the last one x2 of the BETWEEN range (test 286). If $x < x2$, the thesaurus pointer x is incremented by one unit in step 287 for the next iteration of the loop. The loop is over when $x = x2$ in test 286, and the program proceeds to step 266 of figure 42.

The layer $k < n$ normal density processing is detailed in figure 45 in the
 25 case where the thesauruses are arranged as illustrated in figures 25-32. It takes advantage of the fact that, even where $NOk(AD)$ belongs to a layer k rank list associated with a word or macroword of the BETWEEN range, it is useless to access the bitmap segment $HPk(AD)$ if there is a zero in the bit of rank $NOk(AD)$ of the bitmap vector Res obtained in the preceding layer $k+1$.

30 The procedure of figure 45 is comparable to that of figure 44. Steps 280-282 and 285-287 are the same with k substituted for n. However, when a record chain is to be examined ($AD > 0$ in test 282), only the layer k rank value $NOk(AD)$ and the next address $NXk(AD)$ are read as variables N and M in step

WO 02/44943

PCT/IB01/02792

- 60 -

290. The bit $\text{Res}(N)$ of the layer $k+1$ result bitmap Res is then evaluated in test 291. If $\text{Res}(N) = 0$, the rank N is filtered out by jumping directly to step 285. Otherwise ($\text{Res}(N) = 1$), the bitmap segment $\text{HPk}(AD)$ is read in step 293 before proceeding to step 284.

5 With the arrangement of the thesaurus entry coding data, it is noted that the loops of figures 44 and 45 will generally imply the successive reading of contiguous data container records (steps 283 and 290), because each word of index WI has its coding data stored at consecutive addresses AD in the data container, as well as most consecutive words of the BETWEEN range.
10 Therefore, those loops can be executed efficiently by loading blocks of data container records by means of the computer cache memory, thereby reducing the required number of disc accesses. The same consideration applies to the low density data $\text{NO}(AD)$ and $\text{NX}(AD)$ read in step 273 of figure 43.

A further improvement is obtained with the layer $k < n$ normal density processing shown in figure 46, which is made of two successive loops. The first loop, indexed by the thesaurus pointer x , is for determining a temporary rank table noted TNO , which is used to handle the bitmap segments in the second loop. Table TNO has a number of addresses which is at least equal to the number of addresses ADmax of the data container in which the layer k coding data of the current thesaurus (AT, q) are stored. Each entry $\text{TNO}(AD)$ of address AD in the rank table TNO is for containing an integer representing the rank $\text{NOk}(AD)$ if it is useful to access the bitmap segment $\text{HPk}(AD)$, or else a default value (-1) .
20

In the initialization step 279, all entries of the rank table TNO are set to the default value -1 . The first loop is comparable to that of figure 45. When $\text{Res}(N) = 1$ in test 291, the rank N is written at address AD into table TNO in step 295 before substituting M for AD in step 285.
25

When the first loop is over ($x = x2$ in test 286), the program proceeds to the second loop which is initialized with $AD = 1$ in step 301. In each iteration of the second loop, the contents N of the rank table TNO at address AD , read in step 302, are compared with the default value in test 303. If N is a valid rank value ($\neq -1$), the bitmap segment $\text{HPk}(AD)$ is read (step 304) and superimposed, by a bitwise Boolean OR operation, onto the segment $\text{WZ}[N]$ of
30

WO 02/44943

PCT/IB01/02792

- 61 -

the bitmap vector WZ (step 305). If $AD < AD_{max}$ (test 306), the rank table address AD is incremented by one unit in step 307 before coming back to step 302. The second loop is over when $AD = AD_{max}$ in test 306, and the program proceeds to step 266 of figure 42.

5 In addition to filtering out the bitmap segments $HPk(AD)$ that are not worth reading, the procedure illustrated by figure 46, owing to the rank table TNO, groups the read operations in the file containing the layer k bitmap segment data based on the address AD (step 304 in the second loop). Such grouping is not only done word by word but for all words of the BETWEEN
 10 range: when the HPk file is eventually read in the second loop, no more distinction is made between the words for which a rank value has been written into table TNO. This takes maximum advantage of the blockwise access to the HPk file, and provides a very significant advantage because the lower layers, especially layer 1, imply the largest HPk files and the highest numbers of read
 15 operations therein.

Figure 47 shows how the procedure of figure 45 can be adapted when the coding data containers are stored as illustrated in figures 25-30 and 34A-B. The loop has a similar structure. However, since the coding data are accessed from the highest layer n, the address AD read in step 281 is the head address
 20 $AT\_P(q)\_Fn(WI)$ of the record chain in the layer n data container, and when $AD > 0$ in step 282, the rank value $NO_n(AD)$ and next address $NX_n(AD)$ read as variables N and M in step 296 also relate to layer n. After step 296, a filtering function FILT is called in step 297 before substituting M for AD in step 285.

25 A flow chart of this function FILT is shown in figure 48. Its arguments, input when starting its execution in step 500, are as follows (in addition to the attribute name and macroword level which are implicit in figures 47-48):

- a first coding layer index k, corresponding to the first argument of the function FNODE called in step 242 of figure 41;
- 30 - a second coding layer index $k' > k$, with $k' = n$ when the function FILT is called in step 297 of figure 47;

WO 02/44943

PCT/IB01/02792

- 62 -

- $k'-k$ bitmap vectors $Res_{k+1}, Res_{k+2}, \dots, Res_k$, where Res_{k+1} is the layer $k+1$ query result Res . If $k' > k+1$, $Res_{k+2}, \dots, Res_k$ are the bitmap vectors obtained, in step 243 of figure 41, by encoding Res in the higher layers;
- 5 - a layer k' rank N , with $N = NO_n(AD)$ when the function $FILT$ is called in step 297 of figure 47;
- the corresponding record address AD in the layer k' data container; and
- the bitmap vector WZ which is being calculated.

In test 501, it is determined whether the $(N+1)$ -th segment of the
 10 bitmap vector Res_k is only made of zeroes. If so, it is not necessary to read any further coding data relating to the layer k' rank N , so that the execution of the function is terminated in step 502 by returning the bitmap vector WZ .

If the segment $Res_k[N]$ has at least one "1" in test 501, the bitmap segment $HPK(AD)$ is read as segment variable H in step 503, and the
 15 intersection segment $H \text{ AND } Res_k[N]$ is evaluated in test 504. If this intersection segment is only made of zeroes, it is also useless to read any further coding data, and the program directly proceeds to step 502.

If test 504 reveals that $H \text{ AND } Res_k[N]$ has at least one "1", it is necessary to get into the lower layer record chain. Its head address $F(k'-1)(AD)$
 20 is read as variable AD' in step 505, while the layer k' remainder R is initialized to 0 and the layer $k'-1$ rank N' is initialized to $N \times Dk'$. The bitmap segment $H = HPK(AD)$ is scanned in a loop in which its bits $H(R)$ are successively examined (test 506) to ascertain whether the rank $N' = N \times Dk' + R$ should be regarded. If $H(R) = 0$, the rank N' is not in the layer k' coding data of the current
 25 thesaurus entry, so that it is disregarded: R is incremented by one unit in step 507 and if the new R is still smaller than Dk' (test 508), N' is also incremented by one unit in step 509 before proceeding to the next iteration from test 506.

If $H(R) = 1$ in test 506, the $(N'+1)$ -th bit of the vector Res_k is examined
 in test 510 to determine whether the layer $k'-1$ rank N' has been filtered out in
 30 the higher layer processing. If so ($Res_k(N') = 0$), the program jumps to the next position in the layer $k'-1$ record chain by replacing AD' by the next address

WO 02/44943

PCT/IB01/02792

- 63 -

$NX(k-1)(AD')$ in step 511. After step 511, the program proceeds to the above-described step 507.

If $Res_k(N') = 1$ in test 510, the processing depends on whether the coding layer k' is immediately above k (test 512). If $k' = k+1$, the bitmap segment $HPk(AD')$ is read (step 513) and superimposed, by a bitwise Boolean OR operation, onto the segment $WZ[N']$ of the bitmap vector WZ (step 514). If $k' > k+1$ in test 512, the recursive function $FILT$ is called in step 515 with the arguments $k, k'-1, Res_{k+1}, \dots, Res_{k'-1}, N', AD'$ and WZ . After step 514 or 515, the program proceeds to the above-described step 511.

The scanning of the bitmap segment $H = HPk(AD)$ is over when $R = Dk'$ in test 508. The updated bitmap vector WZ is then returned in step 502.

When the coding data containers are arranged as illustrated in figures 34C-D, the scanning of the layer k' bitmap segment in loop 505-509 is replaced by the scanning of the layer k' remainders in the record chain of the layer $k'-1$ coding data container.

The procedure of figures 47-48 has the advantage that the lower layer record chains are accessed only when it is strictly necessary. In particular, it is noted that the loop 282-285 of figure 45 requires the reading of all the layer k ranks (step 290) relating to the current thesaurus entry while it may be already known from the $k+1$ processing that some ranks will be disregarded ($Res(N) = 0$ in test 291). When this occurs in figures 47-48, the rank N is not read in the hard drive (it is not even stored). This advantage is very significant since the lower layers, particularly layer 1, have the largest coding data containers, so that plenty of useless read operation are avoided.

It is noted that the use of a rank table TNO according to figure 46 is quite compatible with the procedure of figures 47-48. The first loop 280-287 of figure 46 is simply replaced by that of figure 47, and steps 513-514 of figure 48 are replaced by writing N' into $TNO(AD')$.

It is noted that the loops of figures 43-47 may cover not only a BETWEEN range in a thesaurus, but generally words and/or macrowords whose coding data are stored in the same data container, and which are combined in an OR type of operation. Instead of running the loops from $x = x1$

WO 02/44943

PCT/IB01/02792

- 64 -

to $x = x2$, an iteration is made for each one of such word or macroword.

For example, if the word and macroword thesauruses for a given attribute share the same data container, the loop may be executed only once for all relevant values of the attribute, i.e. for the sub-tree which, in stage 191 of figure 36, has been substituted for the corresponding node of the query tree.

In addition, such words and/or macrowords may possibly belong to different thesauruses (which requires a suitable labeling of the OR nodes of the query tree). For example, if a query aims at the accidents undergone by a certain client or having a damage amount greater than a given value, and if the client and accident amount thesauruses share the same data containers (as in figures 31-32), the client and accident amount attributes may be examined within the same first loop of figure 46, and the TNO table scanned only once to retrieve all the relevant HP1 segments.

However, it is preferable to have one data container for each thesaurus and each macroword level, as indicated previously. An advantage of this is to reduce the sizes of the rank tables TNO used in the procedure of figure 46.

It is also noted that, when encoding the leaves of the expanded query tree, it is possible to use the word indexes $AT\_P(q)\_WI(x)$ instead of the thesaurus row-ID's x . A list of word indexes is then encoded for each leaf of the expanded query tree. Accordingly, the tree expansion procedure 191 is carried out with reference to the thesaurus word index files, whereas they are not used in the processing of stage 192, which directly calls the record chain head addresses by means of the word indexes. This is useful when the word indexes do not coincide with the thesaurus row-ID's (contrary to figures 25-27), which will normally happen as the database lives.

In the preferred case where separate coding data files are used for each thesaurus word, as in figure 17, the layer n processing of step 264 is similar to that shown in figure 44. The loop is not performed in a common data container (with the loop index AD), but in the individual coding data files $AT\_P(q)\_W\_NOk$ and $AT\_P(q)\_W\_HPk$ (with a loop index i as in figure 33). Optimal disc access is ensured without any thesaurus sorting. The layer $k < n$ processing of step 265 does not need two loops as in figure 46. It may be in accordance with figure 49.

WO 02/44943

PCT/IB01/02792

- 65 -

The first step 310 of the procedure shown in figure 49 consists in allocating the value $AT\_P(q)(x)$ of the word of rank x in the current thesaurus to the word variable W , and in initiating the loop index i to zero. As long as i is lower than the total number $imax(AT, q, W, k)$ of layer k records in the coding data file relating to thesaurus AT , macroword level q and word W (test 311), steps 312-315 are performed. In step 312, the rank $AT\_P(q)\_W\_NOK(i)$ is assigned to the integer variable N . Those rank data are read block by block to minimize the disc accesses. In the following test 313, the bit $Res(N)$ of the layer $k+1$ result bitmap Res is evaluated. If $Res(N) = 1$, the bitmap segment $AT\_P(q)\_W\_HPk(i)$ is read in step 314 and superimposed, by an Boolean OR operation, onto the segment $WZ[N]$ of bitmap vector WZ in step 315, whereby any "1" in $AT\_P(q)\_W\_HPk(i)$ is written at the corresponding position into $WZ[N]$ and any "0" in $AT\_P(q)\_W\_HPk(i)$ leaves unchanged the corresponding bit of $WZ[N]$. The bitmap segment data $AT\_P(q)\_W\_HPk(i)$ are also read by blocks. In step 316, performed after step 315 or when $Res(N) = 0$ in test 313, the loop index i is incremented by one unit before coming back to test 311. When the relevant coding data have been completely examined ($i = imax(AT, q, W, k)$ in test 311), it is determined whether the current word x is the last one $x2$ of the BETWEEN range (test 317). If $x < x2$, the thesaurus pointer x is incremented by one unit in step 318 before coming back to step 310 for the next iteration of the loop. The loop is over when $x = x2$ in test 317.

Figure 50 shows an alternative way of performing the leaf processing of figure 42 (when test 252 is positive), in the case where the coding format of the flat file row-ID lists is specified in the thesaurus index registers, as shown in the right part of figures 25-27.

The initialization step 260A is similar to that 260 of figure 42, except that the rank table TNO is initialized to the default value at the same time. In step 280A, the word index $WI = AT\_P(q)\_WI(x)$ and the corresponding format $F = AT\_P(q)\_FORMAT(x)$ are read from the AT level q thesaurus index register. If F designates "low density" (test 261A), the loop 271-276 depicted in figure 43 is executed in step 262A. Otherwise (F designates "normal density" with n coding layers), the head address $AD = AT\_P(q)\_Fk(WI)$ is read in step 281A to initiate the scanning of a record chain. If we are in the first coding layer

WO 02/44943

PCT/IB01/02792

- 66 -

k = n (test 263A), the loop 282-285 depicted in figure 44 is executed in step 264A. Otherwise, the first loop 282-285 of figure 46 is executed in step 265A. After step 262A, 264A or 265A, the current thesaurus pointer x is compared with the upper bound x2 of the BETWEEN range in test 286A, to be incremented in step 2870A before coming back to step 280A if $x < x2$. When $x = x2$ in test 286A, the table TNO is exploited in step 301A, which is identical to the second loop 301-306 of figure 46, in order to complete the bitmap vector WZ returned in step 266 of figure 42.

QUERY OUTPUT

10 The SQL query further specifies how the data matching the query criteria should be presented in the response. Therefore, the next stage 193 of the query processing (figure 36) is the preparation of the results for their display in stage 194.

Typically, the query defines a list of attributes whose values should be included in the displayed response ("SELECT" and "FROM" clauses in the SQL query, with FROM specifying the relevant data tables and SELECT specifying the relevant columns in those tables).

When a link table of the type shown in figure 9 is stored, the columns of that link table corresponding to the listed attributes are read in the matching rows, identified in the bitmap vector Res output in step 246 of figure 41, in order to obtain the links pointing to the relevant data tables. The attribute values are then retrieved from the data tables for display.

Another possibility is to scan the thesaurus relating to such attribute and to compute the bitwise Boolean AND between the result bitmap vector Res and each encoded bitmap vector of the thesaurus. Every time there is a hit between those vectors (a "1" in the AND output vector), the corresponding thesaurus word will be displayed or otherwise processed. This permits the attribute values of the response to be retrieved without using any link or data table.

30 The AND operations may be performed directly in layer 1. They can also be performed as previously, by decrementing the layer index from $k = n$ to $k = 1$. This requires the layer k results which can be calculated from the layer 1

WO 02/44943

PCT/IB01/02792

- 67 -

bitmap vector Res. The latter option optimizes the disc access by taking advantage of the multi-layer VDG compression scheme.

Such scanning may also be accelerated by taking advantage of the macroword thesauruses. The highest level thesaurus of an attribute is first scanned, and the portions of the lower level thesaurus(es) covered by a given
5 macroword are scanned only if a hit has been observed for the macroword.

Figure 51 shows a procedure suitable for accessing the values to be included in the response for a given attribute AT by scanning the corresponding macroword and/or word thesauruses, which fully takes advantage of both the
10 macroword grouping and the VDG compression scheme.

As before, it is assumed that the attribute AT has a number $Q+1 \geq 1$ of thesauruses indexed by a level parameter q with $0 \leq q \leq Q$, having respective prefix lengths $P(q)$ with $P(0) > P(1) > \dots > P(Q)$, the level parameter $q = 0$ designating the individual word thesaurus, whose prefix length corresponds to
15 the attribute word length. In the notations of figure 45:

- QA is an integer with $0 \leq QA \leq Q$ representing a degree of accuracy expected in the query result; QA is set to 0 for maximum accuracy;
- the thesaurus pointer x_q is a row-ID in the AT thesaurus index register of level q ;
- 20 - for $q \geq QA$, $WZ1_q$ is a bitmap vector which represents a layer q target list of data graph identifiers which match the query criteria and should be examined in connection with the level q thesaurus word x_q . In the initialization step 320, the result bitmap vector Res, output in step 246 of figure 41, is assigned to the vector $WZ1_Q$ which thus represents the flat
25 file row-ID's matching the query criteria;
- for $k > 1$, WZk_q designates a bitmap vector in which each bit of rank N (i.e. the $(N+1)$ -th bit) indicates whether the $(N+1)$ -th segment of $D(k-1)$ bits of $WZ(k-1)_q$ includes at least one "1", in accordance with the VDG compression scheme ($0 \leq q \leq Q$). WZk_q is referred to as a layer k and
30 level q filtering list for $QA \leq q \leq Q$ and $1 \leq k \leq n$. Working zones are

WO 02/44943

PCT/IB01/02792

- 68 -

reserved in RAM 103 for containing the bitmap vectors WZk_q which need not be stored in the hard drive.

In the initialization step 320, the indexes q and x_Q are set to $q = Q$ and $x_Q = 0$, in order to start scanning the highest level thesaurus. In the conversion
 5 step 321, the bitmap vector $WZ1_q$ is processed to provide the corresponding higher layer vectors WZk_Q ($1 < k \leq n$).

The coding layer index k is set to n in step 322, and a function FINTER is called (step 323) to determine the intersection between the integer list represented in the layer k coding data of the thesaurus entry x_q and the filtering
 10 list represented by the bitmap vector WZk_q . The input to this function comprise the coding layer index k , the (macro)word thesaurus level q , the (macro)word index $x = x_q$, and the bitmap vector $WZ = WZk_q$. Its output is another bitmap vector having the same dimension, noted WX , which represents the integer list intersection.

The bitmap vector WX output by the function FINTER called in step
 15 323 is tested in step 324 to determine whether at least one of its bits is "1". If not, the (macro)word pointed to by x_q does not cover any attribute value relevant to the query, so that the thesaurus pointer x_q is incremented by one unit in step 325, and the program comes back to step 322 to examine the next
 20 (macro)word of the level q thesaurus.

If the bitmap vector WX has at least one "1" and if $k > 1$ (following test 326), the layer index k is decremented by one unit in step 327. The next layer processing is then started from step 323.

When $k = 1$ in test 326, $WX \neq 0$ is the bitmap representation of the list
 25 of flat file row-ID's which are represented both in the result bitmap vector Res and in the coding data of the current (macro)word x_q .

If $q > QA$ (test 330), this bitmap vector WX is saved as $WZ1_{q-1}$ in step 331. The row-ID $AT\_P(q)\_FW(x_q)$ of the first "child" of macroword x_q in the lower level thesaurus is then read in the level q thesaurus and assigned as a
 30 starting value of the thesaurus pointer x_{q-1} (step 332). The thesaurus level q is

WO 02/44943

PCT/IB01/02792

- 69 -

then decremented by one unit in step 333, and the lower level processing is started from step 321.

When $q = QA$ in test 330, the word pointed to by x_{QA} (if $QA = 0$), or a word covered by the macroword pointed to by x_{QA} (if $QA > 0$), is an attribute value of a data graph matching the query criteria. In step 335, a certain action is taken based on this word or its thesaurus pointer x_{QA} and the corresponding bitmap vector WX . The latter vector identifies the rows of the flat file which contain the (macro)word x_{QA} in the AT column and which satisfy the query criteria. The type of action depends on the SQL query. Different possibilities will be described further on.

After step 335, the higher level bitmap vectors $WZ1_q$ are updated to remove any "1" present at the same location as in WX . Such a "1" stands for a data graph having the word pointed to by x_{QA} (if $QA = 0$), or a word covered by the macroword pointed to by x_{QA} (if $QA > 0$), as the value of attribute AT; therefore, no other word will have a hit with it, so that it can be removed. To initialize the update, the index q is taken equal to Q in step 336. In step 337, the Boolean operation $WZ1_q \text{ AND NOT } WX$ is performed bit by bit, and the result becomes the updated $WZ1_q$. If the resulting bitmap vector $WZ1_q$ has at least one "1" remaining (test 338), the thesaurus level index q is decremented by one unit in step 339, and step 337 is repeated.

If $WZ1_q$ consists only of zeroes in test 338, it is not necessary to continue the update in the lower levels. If $q < Q$ (test 340), the (macro)word pointed to by x_q does not cover any more attribute value relevant to the query: the thesaurus pointer x_q is incremented in step 341, and the program comes back to step 321 to examine the next (macro)word of the level q thesaurus.

The scanning of the thesauruses for attribute AT is over when $q = Q$ in test 340.

This function FINTER called in step 323 may be in accordance with the flow chart shown in figure 52 when the thesauruses are stored as shown in figures 25-32. It is started in step 350 by loading the above-mentioned input

WO 02/44943

PCT/IB01/02792

- 70 -

arguments k , q , $x (= x_q)$ and $WZ (= WZk_q)$. In step 351, the bitmap vector WX is initialized with zeroes. The program first obtains the word index $WI = AT\_P(q)\_WI(x)$ in step 352, and then the head address $AD = AT\_P(q)\_Fk(WI)$ in step 353 to initiate the scanning of the relevant record chain in the data container.

If the level q thesaurus entry x for attribute AT is stored in the "low density" format (test 354), the processing is as described below with reference to figure 53 (step 355) to obtain the intersection vector WX . If the format is "normal density", the processing depends on whether the program is in the first layer, that is $k = n$ (test 356). The processing of figure 54 is applied if $k = n$ (step 357), and that of figure 55 if $k < n$ (step 358). After step 355, 357 or 358, the execution of function $FINTER$ is terminated in step 359 by returning the bitmap vector WX .

The low density processing of figure 53 has a loop in which each iteration begins by comparing the address AD with the end-of-chain value (0) in test 360. If $AD > 0$, there remains at least one item to be examined in the record chain, so that the flat file row-ID value $NO(AD)$ and the next address $NX(AD)$ are read as variables N and M , respectively, in step 361. The Euclidean division of N by Δk is made in step 362 to obtain the layer $k-1$ quotient (rank) N' . If $WZ(N') = 1$ in the following test 363, a "1" is written into bit $WX(N')$ of the bitmap vector WX (step 364). After step 364, or if $WZ(N') = 0$ in test 363, the variable M is substituted for AD in step 365 before coming back to test 360. The low density processing for the current (macro)word is over when the record chain has been completely examined ($AD = 0$ in test 360), and the program proceeds to step 359 of figure 52.

The layer n normal density processing of figure 54 has a similar loop in which each iteration begins, in step 370, by comparing the address AD with the end-of-chain value (0). If $AD > 0$, the layer n rank value $NO_n(AD)$ and the next address $NX_n(AD)$ are read as variables N and M , respectively, in step 371. If the segment of rank N in the bitmap vector WZ has at least one "1" ($WZ[N] \neq 0$ in the following test 372), the bitmap segment $HP_n(AD)$ is read (step 373) and combined with the bitmap segment $WZ[N]$ in a bitwise Boolean AND operation

WO 02/44943

PCT/IB01/02792

- 71 -

to provide the segment $WX[N]$ of the bitmap vector WX (step 374). After step 374, or if $WZ[N] = 0$ in test 372, the variable M is substituted for AD in step 375 before coming back to test 370. The layer n normal density processing for the current (macro)word is over when the record chain has been completely examined ($AD = 0$ in test 370), and the program proceeds to step 359 of figure 52.

The layer $k < n$ normal density processing is advantageously made of two successive loops (figure 55). The first loop is for determining a temporary rank table TNO , which is used to handle the bitmap segments in the second loop, like in the procedure described previously with reference to figure 46. Table TNO has a number of addresses which is at least equal to the number of addresses AD_{max} of the data container in which the layer k coding data of the current thesaurus (AT, q) are stored. Each entry $TNO(AD)$ of address AD in the rank table TNO is for containing an integer representing the rank $NOK(AD)$ if it is useful to access the bitmap segment $HPk(AD)$, or else a default value (-1) . Such access is useless if $NOK(AD)$ does not belong to the layer k rank list associated with the current (macro)word x_q , or if there are only zeroes in the segment of rank $NOK(AD)$ in the bitmap vector $WZ = WZ_{k+1}$.

In the initialization step 380, all entries of the rank table TNO are set to the default value -1 . Each iteration of the first loop begins in step 381 by comparing the address AD with the end-of-chain value (0) . If $AD > 0$, the layer k rank value $NOK(AD)$ and the next address $NXk(AD)$ are read as variables N and M , respectively, in step 382. The segment $WZ[N]$ of rank N in the bitmap vector WZ is examined in test 383. If that segment $WZ[N]$ has at least one "1" ($WZ[N] \neq 0$ in test 383), the rank N is written at address AD into table TNO in step 384 before substituting M for AD in step 385 and coming back to test 381 to examine the next record of the chain. Otherwise ($WZ[N] = 0$), the rank N is filtered out by jumping directly to step 385.

The first loop is over when the record chain has been completely examined ($AD = 0$ in test 381). The program then proceeds to the second loop 386-391. In each iteration of the second loop, the contents N of the rank table TNO at address AD , read in step 387 after having incremented AD in step 386,

WO 02/44943

PCT/IB01/02792

- 72 -

are compared with the default value in test 388. If N is a valid rank value ($\neq -1$), the bitmap segment HPk(AD) is read (step 389) and combined with the bitmap segment WZ[N] in a bitwise Boolean AND operation to provide the segment WX[N] of rank N in the bitmap vector WX (step 390). If AD < ADmax (test 391),
 5 the rank table address AD is incremented by one unit in step 386 when starting the next iteration. The second loop is over when AD = ADmax in test 391, and the program proceeds to step 359 of figure 52.

The scanning of the thesauruses as explained with reference to figures 51-55 has a number of significant advantages:

- 10 - it does not require any access to the original data tables. Therefore it is not compulsory to maintain the data tables in memory. Even when they are stored, they will often be accessible through a relatively low software interface, such as ODBC. The scanning method advantageously circumvents that interface;
- 15 - it is very efficient in terms of disc accesses, because it takes advantage of the record grouping in the coding data container. The procedures of figures 53-55 are respectively similar to those of figures 43, 44 and 46 regarding the disc accesses, and they provide the above-described advantages in this respect;
- 20 - the procedure of figure 51 is also very efficient owing to the filtering achieved by the updating of the bitmap vectors WZ1_q (loop 336-339). This filtering takes advantage of the fact that each flat file row has a unique value (possibly *Null*) for each attribute. It avoids plenty of useless operations to read coding data pertaining to subsequent thesaurus words and macrowords which would not provide hits in the lowest layer
 25 (because the hit in the higher layer would be due to a flat file row-ID corresponding to an already considered thesaurus word).

Figure 56 shows how the procedure of figure 51 can be adapted when the coding data containers are stored as illustrated in figures 25-30 and 34A-B.
 30 The above-described function FINTER is replaced by a recursive function FFILT illustrated by figure 57. Accordingly, the loop 322-327 is replaced by a loop 590-593 after executing steps 351-353 as in figure 52 (with k = n). If the

WO 02/44943

PCT/IB01/02792

- 73 -

resulting intersection bitmap WX is made of zeroes only (test 324), x_q is incremented in step 325 before coming back to step 351 for the next (macro)word of the current level q thesaurus range. If WX has at least one "1" in test 324, the program proceeds to step 330 as described before. Otherwise, the procedure of figure 56 is the same as that of figure 51.

Each iteration in the loop 590-593 begins by comparing the address AD with the end-of-chain value (0) in test 590. If $AD > 0$, the layer n rank value $NO_n(AD)$ and the next address $NX_n(AD)$ are read as variables N and M, respectively, in step 591. Afterwards, the filtering and intersection function FFILT is called in step 592 before substituting M for AD in step 593. The computation of the intersection list WX for the current (macro)word is over when the layer n record chain has been completely examined ($AD = 0$ in test 590), and the program proceeds to test 324 as indicated hereabove.

A flow chart of this function FFILT is shown in figure 57. Its arguments, input when starting its execution in step 600, are as follows:

- a coding layer index k, with $k = n$ when the function FFILT is called in step 592 of figure 56;
- k bitmap vectors $WZ1_q, WZ2_q, \dots, WZk_q$ as obtained in step 321 of figure 56;
- a layer k rank N, with $N = NO_n(AD)$ when the function FFILT is called in step 592 of figure 56;
- the corresponding record address AD in the layer k data container; and
- the intersection bitmap vector WX which is being calculated.

In test 601, it is determined whether the segment of rank N of the bitmap vector WZk_q is only made of zeroes. If so, it is not necessary to read any further coding data relating to the layer k rank N, so that the execution of the function is terminated in step 602 by returning the bitmap vector WX.

If the segment $WZk_q[N]$ has at least one "1" in test 601, the bitmap segment $HPk(AD)$ is read as segment variable H in step 603, and the intersection segment $H \text{ AND } WZk_q[N]$ is evaluated in test 604. If this intersection segment is only made of zeroes, it is also useless to read any further coding data, and the program directly proceeds to step 602.

WO 02/44943

PCT/IB01/02792

- 74 -

If test 604 reveals that $H \text{ AND } WZk_q[N]$ has at least one "1", it is necessary to get into the lower layer record chain. Its head address $F(k-1)(AD)$ is read as variable AD' in step 605, while the layer k remainder R is initialized to 0 and the layer $k-1$ rank N' is initialized to $N \times Dk$. The bitmap segment $H = HPk(AD)$ is scanned in a loop in which its bits $H(R)$ are successively examined (test 606) to ascertain whether the rank $N' = N \times Dk + R$ should be regarded. If $H(R) = 0$, the rank N' is not in the layer k coding data of the current thesaurus entry, so that it is disregarded: R is incremented by one unit in step 607 and if the new R is still smaller than Dk (test 608), N' is also incremented by one unit in step 609 before proceeding to the next iteration from test 606.

If $H(R) = 1$ in test 606, the bit of rank N' of the vector WZk_q is examined in test 610 to determine whether the layer $k-1$ rank N' is in the result list. If not ($WZk_q(N') = 0$), the program jumps to the next position in the layer $k-1$ record chain by replacing AD' by the next address $NX(k-1)(AD')$ in step 611. After step 611, the program proceeds to the above-described step 607.

If $WZk_q(N') = 1$ in test 610, the processing depends on whether the coding layer k is immediately above 1 (test 612). If $k = 2$, the bitmap segment $HP1(AD')$ is read (step 613) and combined with the bitmap segment $WZ1_q[N']$ in a bitwise Boolean AND operation to provide the segment $WX[N']$ of rank N' in the bitmap vector WX (step 614). If $k > 2$ in test 612, the recursive function $FFILT$ is called in step 615 with the arguments $k, WZ1_q, \dots, WZ(k-1)_q, N', AD'$ and WX . After step 614 or 615, the program proceeds to the above-described step 611.

The scanning of the bitmap segment $H = HPk(AD)$ is over when $R = Dk$ in test 608. The updated bitmap vector WX is then returned in step 602.

It is noted that the use of a layer 1 rank table TNO (as in figure 55) is quite compatible with the procedure of figures 56-57. The records of the table TNO are initialized with the default value in step 351; steps 613-614 of figure 57 are replaced by writing N' into $TNO(AD')$; and when $AD = 0$ in test 590, table TNO is scanned as in loop 386-391 of figure 55.

A further optimization of the procedure of figure 51 or 56 can be achieved when the stored thesaurus data include files organized as illustrated

WO 02/44943

PCT/IB01/02792

- 75 -

in figures 58-61. For each thesaurus, a table of the type shown in figures 58-60 is stored, to associate each possible value of the layer n rank NO_n with a record chain head address F_AD' in an additional data container as shown in figure 61. The latter data container contains the same layer n bitmap segment data $HP'2 = HP2$ as that of figure 32 or 34A, but the links $NX'2$ define record chains which pertain to the same layer n rank rather than to the same thesaurus entry. The data container of figure 61 is thus obtained by sorting that of figure 32 or 34A based on the $NO2$ column, deleting the $NO2$, $NX2$ and $F1$ columns, and adding a column $NX'2$ to contain the next addresses in the record chains based on $NO2$ and a further column PTR where the thesaurus indexes x to which the record pertain is written. For each rank $NO2$ the head address of the chain is memorized in $F\_AD'(NO2)$.

Before starting the procedure of figure 51 or 56, or after every iteration of step 321, the pre-filtering treatment shown in figure 62 is applied to mark thesaurus entries that will not be read for the reason that their layer n ranks are not in the layer n coding data of the matching data graph identifier list. The marking is done by means of a table T_q for a macroword level q , which has one bit $T_q(x_q)$ for each level q thesaurus pointer x_q . Those bits, as well as the layer n rank $N = NO_n$ are initialized to zero in step 620 of figure 62. If the segment of rank N of WZn_q is only made of zeroes (test 621), test 622 is performed to determine whether the highest possible layer n rank $NO_{n_{max}}$ has been reached. If not, N is incremented in step 623 and test 621 is repeated. When $WZn_q[N] \neq 0$ in test 621, the head address $F\_AD'(N)$ is read as variable AD' in step 624 and compared to the end-of-chain value (0) in test 625. If $AD' = 0$, the program proceeds to step 622. Otherwise, the bitmap segment $HP'n(AD')$ and the corresponding next address value $NX'n(AD')$ are read as variables H' and M' , respectively, in step 626. If H' and $WZn_q[N]$ have no "1" in common (test 627), M' is substituted for AD' in step 628, and the next iteration is started from test 625. If there is at least one "1" in the bitwise Boolean AND combination of H' and $WZn_q[N]$ in test 627, the thesaurus pointer $x_q = PTR(AD')$ is read in the last column of figure 61, and a "1" is written in the corresponding location of

WO 02/44943

PCT/IB01/02792

- 76 -

table T_q before proceeding to step 628.

After that, as shown in figure 63, the loop 322-327 of figure 51, where a relevant bitmap vector WX is calculated, is completed by an initial filtering step 640 where the bit $T_q(x_q)$ is tested. This test 640 is also performed after having
 5 incremented x_q in step 325. If $T_q(x_q) = 1$ in test 640, the program proceeds to step 322 as described before. If $T_q(x_q) = 0$, it proceeds directly to step 325, thereby avoiding the computation of an intersection list WX that will be empty.

The same filtering step 640 can be performed before step 351 in figure 56.

10 The function FINTER illustrated in figures 52-55 is readily adapted to the case where separate coding data files are used for each thesaurus word, as in figure 17. Steps 352-353 of figure 52 are replaced by the allocation of the value $AT\_P(q)(x)$ to the word variable W , and by the initialization of the loop index i to zero. The low density processing of step 355 and the layer n normal
 15 density processing of step 357 are similar to those shown in figures 53 and 54. The loop is not performed in a common data container (with the loop index AD), but in the individual coding data files (with a loop index i as in figure 33). The layer $k < n$ processing of step 358 does not need two loops as in figure 55. It may be in accordance with figure 64.

20 In the procedure shown in figure 64, steps 395-399 are performed as long as the loop index i is lower than the total number $imax(AT, q, W, k)$ of layer k records in the coding data file relating to thesaurus AT , macroword level q and word W (test 394). In step 395, the rank $AT\_P(q)\_W\_NOK(i)$ is assigned to the integer variable N . In the following step 396, the segment $WZ[N]$ of rank N
 25 in the bitmap vector WZ is tested. If $WZ[N]$ has at least one "1" ($WZ[N] \neq 0$), the bitmap segment $AT\_P(q)\_W\_HPk(i)$ is read (step 397) and combined with the bitmap segment $WZ[N]$ in a bitwise Boolean AND operation to provide the segment $WX[N]$ of rank N in the bitmap vector WX (step 398). In step 399, performed after step 398 or when $WZ[N] = 0$ in test 396, the loop index i is
 30 incremented by one unit before coming back to test 394. The loop is over when the relevant coding data have been completely examined, i.e. when $i = imax(AT, q, W, k)$ in test 394.

WO 02/44943

PCT/IB01/02792

- 77 -

The above-described procedure may involve different types of action in step 335 of figure 51 or 56, based on features of the SQL query.

In a relatively simple type of SQL query, a list of values of one attribute is required (e.g. name all clients who meet certain criteria). In such a case, the scanning of figure 51 or 56 is performed only in the thesaurus(es) relating to that attribute, with $QA = 0$, and the action of step 335 may simply be to read the word $AT(x_0)$ which is in position x_0 in the individual word thesaurus (in fact, if the coding data are stored as illustrated in figure 17, the word $AT(x_0)$ has been read just before) and to write this word $AT(x_0)$ into an output table, or print it out. It is observed that the word list thereby produced is automatically sorted, in the ascending order. If the reverse order is required, the thesaurus may be scanned in the opposite direction.

If the SQL query has a DISTINCT keyword in the SELECT clause regarding the attribute AT, there is one output of the word $AT(x_0)$ in step 335. If not, or if the SELECT clause has the keyword ALL, there may be one output of the word $AT(x_0)$ for each non-zero bit of WX in step 335. Those non-zero bits may also be counted to provide the number of occurrences of the word $AT(x_0)$ in the matching data graphs.

If the values of the attribute are required with a reduced accuracy, the thesaurus may be scanned as shown in figure 51 or 56 with $QA > 0$, thereby avoiding memory accesses to obtain irrelevant details from the level q thesauruses with $q < QA$. For example, if a date attribute is required expressed in years, the scanning of figure 51 or 56 may be stopped at the level QA corresponding to a truncation length of 4.

The SQL query frequently requires several attributes in the SELECT and FROM clauses. In order to maintain the connections between the attribute values belonging to the same data graph, some form of indexing is needed. A possibility is to reserve in RAM 103 a working zone for containing an output table having as many rows as in the virtual flat file and respective columns for receiving the attribute values of the result lists. The memory locations of the output table are initialized with a default value. The above-mentioned attribute

WO 02/44943

PCT/IB01/02792

- 78 -

values $AT(x_0)$, or their prefixes $AT(x_{QA})$ if $QA > 0$, are written into the output table in the occurrences of step 335 shown in figure 51 or 56. Such write operation in step 335 is made into any row of the output table indicated by a non-zero bit of the bitmap vector WX . The output data are eventually produced by eliminating the empty rows from the output table (the rows that still contain the default value).

Figure 65 shows how step 335 is developed in such a case, to write the word $W = AT(x_{QA})$ where appropriate in the column OT_AT of the output table. The row pointer j is initialized to zero in step 400, and the word W is loaded (if it has not been before). Every time the bit $WX(j)$ is 1 (test 401), the word W is written into row j and column AT of the output table (step 402). The row pointer j is then compared to its maximum value $jmax$ in test 403 and incremented if $j < jmax$ (step 404). The program has finished the action of step 335 when $j = jmax$ in test 403.

15 Example 4:

We consider the query criteria of Example 3 and assume that the attributes requested for display are accident date, client name and policy date. In Example 3, discussed with reference to figures 37-38, the bitmap of the matching data graphs (output in step 246 of figure 41) is $Res = 101100001000$, as may be checked in figure 8. In this example, figure 66 shows the contents of the output table as described hereabove.

The above-mentioned output table may be too big to be conveniently reserved in RAM 103. In real databases, the number of rows in the virtual flat file is relatively high (e.g. millions) and if there are too many characters in one row of the output table (because there are too many attributes to be included or because some of them use a relatively high number of characters), the output table may become prohibitively big. There are several solutions to deal with this potential problem.

One of them is to write the thesaurus row-ID's x_{QA} (integers) into the output table instead of the (macro)words $AT(x_{QA})$ in step 402 of figure 65. Once all the relevant thesauruses have been scanned, the non-empty output

WO 02/44943

PCT/IB01/02792

- 79 -

table rows are selected to retrieve the attribute values from the thesaurus row-ID's. This reduces the breadth of the columns of the output table since the words $AT(x_{QA})$ often require much more characters.

Figure 67 shows the contents of such an output table in the case of Example 4, the thesauruses being sorted as in figures 10A-G.

Another solution, alternative or cumulative, is to use an index in RAM 103, to associate an integer address with each data graph or flat file row-ID. A default address is initially assigned to all the data graphs. When one of them is designated for the first time by a "1" in the corresponding bit of WX in step 335 (i.e. when scanning the first thesaurus), it is allocated a new address obtained by incrementing a counter. This address is retrieved from the index when the data graph is again designated in the scanning of the subsequent thesaurus(es). This integer address is a row-ID in an output table stored in RAM 103, which has a reduced number of rows where the attribute values or prefixes $AT(x_{QA})$, or their thesaurus row-ID's x_{QA} , are written. The non-empty rows are consecutive and hence the total number of rows can be significantly reduced. This compressed output table is eventually read out to display the results.

Figure 68 shows the contents of such index and output table, containing thesaurus row-ID's, in the case of Example 4.

Figures 69 and 70 show how step 335 is developed when scanning the first thesaurus and the subsequent thesaurus(es), respectively. The steps 400, 401, 403, 404 indicated by the same reference numerals are identical to those of figure 65. In figure 69, when the bit $WX(j)$ is 1, the counter value m (initialized to 0 in step 320 of figure 51 or 56) is allocated to the index $IND(j)$ for row j (step 410), the thesaurus pointer x_{QA} (or word $W = AT(x_{QA})$) is written into row j and column AT of the output table (step 411), and the counter value m is incremented (step 412). When the scanning the first thesaurus is over, m represents the number of matching data graphs. In figure 70, when the bit $WX(j)$ is 1, the index $IND(j)$ for row j is retrieved as pointer m' (step 413) and the thesaurus pointer x_{QA} (or word W) is written into row m' and column AT of the output table (step 414).

WO 02/44943

PCT/IB01/02792

- 80 -

The output table is easily sorted based on the contents of its columns when the SQL query has GROUP BY, ORDER BY or similar clauses. Such sorting operation may be performed hierarchically with reference to a plurality of attributes. The most significant attribute in the hierarchy is preferably
5 subjected to the first thesaurus scanning as shown in figure 51 or 56 so that the first sorting criterion will be automatically fulfilled when constructing the output table. The sorting based on the remaining attributes is done within each portion of the output table that has common values for the previous attribute(s).

The sorting is particularly simple when the columns of the output table
10 contain thesaurus row-ID's x_{QA} , as in figure 68, because it only involves sorting integer lists.

It has been indicated before that for certain attributes, in particular numerical fields, the explicit attribute values may be stored in the link table (if there is a link table). The output table of the type illustrated in figure 66, 67 or
15 68 need not have a column for such attribute. If the attribute is to be displayed or otherwise exploited, its values can be retrieved from the link table in the rows corresponding to (i.e. having the same row-ID as) the non-empty rows of the output table (figures 66-67) or the valid pointers in the output table index (figure 68).

20 SQL queries may also require calculations to be made on attribute values of the matching data records, particularly in data warehousing applications. Such calculations can be performed from the data of an output table of the type illustrated in figure 66, 67 or 68.

Example 5:

25 From Example 4, we assume that the (arithmetic) mean value of the time difference between the accident date and the policy date is requested, expressed as a number of days. For each non-empty row of the output table, the program computes the difference, in number of days, between the first and third column. Those differences are accumulated and the result is divided by
30 the number of non-empty rows (4) to obtain the desired mean value.

In fact, this mean value can be computed with an output table reduced to only one memory location: when scanning the accident date thesaurus, the

WO 02/44943

PCT/IB01/02792

- 81 -

attribute value expressed as a number of days from an arbitrary reference day is multiplied by the number of non-zero bits in WX in step 335 of figure 51 or 56 and added to an accumulation variable V (initialized to 0 in step 320) stored in the memory location of the reduced output table; then, when scanning the
 5 policy date thesaurus, the attribute value expressed as a number of days from the same reference day is multiplied by the number of non-zero bits in WX in step 335 and subtracted from V in step 335; finally, the resulting V is divided by the number of non-zero bits in the result bitmap Res to provide the desired mean value.

10 However, an output or computation table having more than one memory location is often useful in RAM 103 for that sort of calculations, in particular in cases where the desired quantity is not linear with respect to the attribute values (e.g. if the quadratic or geometric, rather than arithmetic, mean value is requested in Example 5).

15 A computation table is a particular case of output table, and it has a structure similar to that of the output table described hereabove. It may have as many rows as in the virtual flat file (as the output tables of figures 66-67). Alternatively, it may be associated with an index identical to that of figure 68. It may also have only one row, as in the above example of the output table
 20 having one memory location. Each column of the computation table is for containing values of an operand used in the calculation to be made. Depending on the complexity of the calculation, one or more columns may be needed, but in most cases one column will be sufficient.

The attribute whose values are involved in the calculation have their
 25 thesauruses scanned successively, as described with reference to figure 51 or 56. Step 335 may be developed as shown in figure 71 in the case of a computation table CT having a single column and as many rows as in the virtual flat file (when there is an index, it can be handled as in figures 69-70). In figure 71, steps 400, 401, 403 and 404 are identical to those of figure 65. When
 30 the bit WX(j) is 1 in step 401, the contents CT(j) of the computation table in row j is allocated to the operand Y in step 416, and then a function f of the operand Y and of the current (macro)word $W = AT(x_{QA})$ is calculated and saved as the

WO 02/44943

PCT/IB01/02792

- 82 -

new contents CT(j) in step 417.

The mathematical function f is selected on the basis of the calculation to be performed and of the thesaurus being scanned. Referring again to Example 5, when the accident date is first scanned, the function $f(Y,W)$ may be the transformation of the date W expressed in the format $yyyy\ mm\ dd$ into a number of days from a reference day (it is thus a function of W only); when the policy date thesaurus is scanned, the function $f(Y,W)$ may consist in applying the same transformation to the date W and subtracting the result from Y . Afterwards, the mean value (arithmetic, quadratic, geometric, ...) of the non-empty rows of CT is calculated to provide the desired output result. Other kinds of global calculation can be performed from the columns of the computation table, for example statistical, financial or actuarial calculations.

The macrowords are advantageously used in this type of calculation if the desired accuracy is lower than that afforded by the individual words of at least one of the attributes involved.

VIRTUAL FLAT FILE PARTITIONING

For large systems, it is often advantageous to partition the virtual flat file into several portions or blocks each consisting of a determined number of rows. The data graphs are distributed into the blocks based on their identifiers (flat file row-ID's).

Preferably, each thesaurus is divided into corresponding thesaurus sections, whereby each section has entries whose flat file row-ID lists are included in the corresponding virtual flat file block. The complete flat file row-ID list associated with one word assigned to an attribute is the union of the lists represented in the entries of the corresponding thesaurus sections for that word. Accordingly the complete flat file row-ID lists of the thesaurus entries are subjected to the same partitioning as the virtual flat file: they are split into sub-lists corresponding to the thesaurus sections.

The thesaurus index file for an attribute may be common to all the sections. A separate index file may also be provided for each section.

For each one of the blocks, steps 191-193 of the processing of a SQL query (figure 36) are performed as described hereabove with reference to

WO 02/44943

PCT/IB01/02792

- 83 -

figures 38-71. The results thus obtained are merged to display the response.

The processing of the query with respect to the different blocks may be performed sequentially or in parallel.

In a sequential processing, RAM availability for optimal processing speed can be effectively controlled. Even though the cost of RAM circuits is not currently considered to be critical, a given machine has a certain amount of available RAM capacity and this is a limitation to reserve RAM space for the above-described output or computation tables. When the limitation is likely to be encountered, partitioning the virtual flat file directly reduces the size of those tables (jmax in figures 65 and 69-71).

Accordingly, the use of a particular machine to carry out the invention will dictate the choice of jmax, that is the block size. The virtual flat file blocks are dimensioned based on the selected size parameter, and the corresponding thesaurus sections are constructed one section after the other as indicated with reference to steps 122-126 of figure 19.

Such dimensioning of the query processing engine enables to use optimal algorithms at all stages while avoiding the need to swap intermediary data between RAM 103 and hard drive 105.

A further acceleration is achieved when parallel processing is used. The query processing is distributed between several processors, one for each virtual flat file block.

A possible architecture of the parallel query processing engine is illustrated in figure 72, in the particular case where all blocks have the same size jmax. A number M of matching units 700 are connected to a query server 701 through a communication network 702. Each matching unit 700 may be a processor system of the type shown in figure 18. It has a storage device 703 such as a hard drive for storing the thesaurus sections associated with the block. If a link table of the type shown in figure 9 is used, it is partitioned into blocks in the same manner as the virtual flat file, and each block is stored in the corresponding matching unit. The server 701 provides the man-machine interface. It translates the query criteria of the SQL WHERE clause into trees of the type shown in figure 37, which are provided to the M matching units 700 along with a description of the desired output. Each of the units 700 does its

WO 02/44943

PCT/IB01/02792

- 84 -

part of the job according to steps 191-193 of figure 36 and returns its response to the server 701. The latter compiles the results from the different matching units to provide the overall response to the user. In order to perform the analysis of step 191, each matching unit 700 uses its thesaurus sections.

5 Alternatively, the analysis of the query criteria could be executed centrally by the server 701 by means of global thesauruses, each global thesaurus being common to all the (macro)words and having M columns for containing pointers to identifier sub-lists in the M storage units 703. At the end of the analysis stage, the relevant pointers are addressed to the matching units
10 700 for their execution of steps 192-193.

An update server 704, which may be the same machine as the query server 701, is also connected to the network 702 to create and maintain the VDG's relating to the different blocks. It monitors the changes made in the data tables of the RDBMS and routes thesaurus update commands to the units 700
15 in order to make the necessary changes in the thesaurus sections.

The above-described parallel system is readily extended when the number of data graphs becomes close to the current maximum ($M \times j_{\max}$ in the illustration of figure 72). This requires the addition of a further matching unit to deal with a new virtual flat file block, whose size may be the same as or
20 different from the previous blocks, and a reconfiguration of the routing and result compilation functions in the servers 701, 704. The reconfiguration is completely transparent to the previously existing matching units. Therefore, increasing the system capacity can be done at a minimum cost. It does not even require to shut down the system.

WO 02/44943

PCT/IB01/02792

- 85 -

CLAIMS

1. A method of organizing information in a database system, wherein a group of attributes is defined and words of a collection of data are assigned to said attributes, wherein the group of attributes is divided into a plurality of sub-
5 groups each associated with a respective data table, each data table having a column for each attribute of the associated sub-group and rows for containing data table records comprising at least one word assigned to an attribute of the associated sub-group, wherein links are defined between the data tables records, each link having a target table and a corresponding source table
10 having a link column containing link values each designating a record of said target table, whereby each of said link values represents a link between the record of the source table including said link value and the record of the target table designated by said link value, the method comprising the steps of:
 - allocating respective identifiers to data graphs, wherein each data graph
15 represents related attribute values respectively assigned to the attributes of said group, wherein each attribute value of a data graph is either a default value or a word of said collection of data, and wherein the words of each data graph are from linked data table records;
 - storing a plurality of word thesauruses respectively associated with
20 attributes of said group, wherein for each word assigned at least once to an attribute in the collection of data, the word thesaurus associated with said attribute has a respective entry containing said word; and
 - storing data representing data graph identifier lists respectively
25 associated with the word thesaurus entries, wherein the data graph identifier list associated with a thesaurus entry relating to a word assigned to an attribute includes any identifier allocated to a data graph having said word assigned to said attribute.
2. A method according to claim 1, wherein the entries of each word
30 thesaurus associated with an attribute of said group are sorted based on the words assigned to said attribute.

3. A method according to claim 1 or 2, further comprising the step of storing a link table having a plurality of rows respectively associated with the data graphs and a plurality of columns respectively associated with the attribute sub-groups, wherein each row of the link table contains, in each one of the
5 columns, either a value indicating that each attribute value represented in the data graph associated with said row and assigned to an attribute of the sub-group associated with said one of the columns is a default value or a link value for retrieving at least one stored word of the collection of data represented in the data graph associated with said row and assigned to an attribute of the sub-
10 group associated with said one of the columns.
4. A method according to claim 3, wherein said data tables are stored, and wherein each link value contained in the column of the link table associated with an attribute sub-group comprises data for identifying a row of the data table associated with said sub-group.
- 15 5. A method according to claim 3 or 4, wherein the link table has at least one column associated with a sub-group consisting of one attribute, each link value contained in said column being a word assigned to said one attribute in the collection of data.
6. A method according to claim 3, wherein the link table has at least
20 one column associated with a sub-group consisting of one attribute, each link value contained in said column being a pointer to an entry of the word thesaurus associated with said one attribute.
7. A method according to any one of the preceding claims, wherein each word thesaurus associated with an attribute of the group to which the
25 default value is assigned in at least one of the data graphs further has an entry for the default value, whereby one of said data graph identifier lists is associated with said thesaurus entry for the default value and includes any identifier allocated to a data graph having said default value assigned to said attribute.

WO 02/44943

PCT/IB01/02792

- 87 -

8. A method according to claim 7, wherein the step of allocating identifiers to the data graphs comprises:
- for any target table record which is not designated in the link column of the corresponding source table, inserting a dummy record in an additional row of said source table, having a link value designating said target table row in the link column and a default value in each attribute column, such insertion of dummy records being repeated until any record of any target table is designated in the link column of the corresponding source table;
 - initializing the data representing data graph identifier lists, to represent empty lists; and
 - processing the records of at least one root table which is not the target table for any link, the processing of each root table record comprising:
 - constructing a respective data graph consisting of related attribute values of said root table record and of corresponding records of the other data tables successively determined on the basis of the contents of the link columns in said records;
 - allocating an available identifier to said data graph; and
 - for each attribute, selecting the entry of the associated word thesaurus relating to the value assigned to said attribute in said data graph and updating the data representing the data graph identifier list associated with the selected entry to add the identifier allocated to said data graph to said list.
9. A method according to claim 8, wherein the processing of each root table record further comprises:
- generating a row in a link table having a plurality of columns respectively associated with the data tables, wherein said row of the link table contains, in each column associated with a data table, either a value indicating that the data graph has been constructed from a dummy record with respect to said data table or a link value designating a row of said data table

WO 02/44943

PCT/IB01/02792

- 88 -

- where the attribute values represented in the data graph are stored,
said link table being used for retrieving from the data tables words of the collection of data represented in the data graphs.
- 5 10. A method according to claim 9, further comprising the step of deleting the dummy records from the data tables.
11. A method according to claim 9 or 10, further comprising the step of deleting the link columns from the data tables.
12. A method according to any one of the preceding claims, further
10 comprising the step of storing at least one macroword thesaurus associated with an attribute of said group and with a prefix length shorter than a word length of said attribute, having a respective entry for each prefix value having said prefix length and matching a corresponding prefix of at least one word assigned at least once to said attribute in the collection of data, said entry of
15 the macroword thesaurus being associated with a data graph identifier list including each identifier allocated to a data graph having, assigned to said attribute, a word whose corresponding prefix matches said prefix value.
13. A method according to claim 12, wherein at least one attribute has a plurality of stored macroword thesauruses, associated with different prefix
20 lengths.
14. A method according to claim 12 or 13, wherein the entries of each macroword thesaurus associated with an attribute are sorted based on the prefix values, and the entries of the word thesaurus associated with said attribute are sorted based on the words assigned to said attribute.
- 25 15. A method according to claim 14, wherein at least one attribute has Q stored macroword thesauruses associated with respective prefix lengths shorter than a word length of said attribute, Q being a positive integer, each of said Q macroword thesauruses having a thesaurus level parameter q such that $1 \leq q \leq Q$, the prefix length being a decreasing function of the level parameter

WO 02/44943

PCT/IB01/02792

- 89 -

q, the word thesaurus associated with said attribute having a level parameter $q = 0$, wherein each macroword thesaurus having a level parameter $q \geq 1$ contains, in each entry provided for a level q prefix value, data designating the entry of the level q-1 thesaurus which corresponds to the lowest or highest
5 level q-1 macroword or word whose corresponding prefix matches said level q prefix value.

16. A method according to any one of the preceding claims, wherein at least one thesaurus entry comprises data for pointing to at least one memory region where coding data representing the data graph identifier list associated
10 with said entry are stored.

17. A method according to claim 16, wherein said memory region comprises a file individually allocated to said thesaurus entry.

18. A method according to claim 17, wherein said data for pointing to at least one memory region comprise the attribute value for which said thesaurus
15 entry is provided, said attribute value being part of a respective file name used for accessing each file allocated to said thesaurus entry.

19. A method according to claim 16, wherein said memory region comprises a portion of a data container having a plurality of records each having a next address field, said portion being defined as a record chain in said
20 data container, the chains being defined by means of the next address fields, and wherein said data for pointing to at least one memory region comprise a respective address of a first record of a chain in said data container.

20. A method according to claim 19, further comprising the step of grouping the records stored in the data container so that the records of each
25 chain have contiguous addresses.

21. A method according to claim 19 or 20, wherein said data container is individually allocated to a thesaurus.

22. A method according to claim 19 or 20, wherein said data container is shared by a plurality of thesauruses.

WO 02/44943

PCT/IB01/02792

- 90 -

23. A method according to any one of claims 19 to 22, wherein the thesaurus associated with an attribute of said group has an index register where the thesaurus entries are sorted based on the words assigned to said attribute, each entry including a word index for pointing to a row of an auxiliary
5 table, and wherein each row of the auxiliary table contains an address in the data container of a first record of a chain.
24. A method according to any one of claims 19 to 23, wherein said data for pointing to at least one memory region comprise a respective address of a last record of a chain in said data container.
- 10 25. A method according to any one of claims 16 to 24, wherein said coding data representing a data graph identifier list comprises integers respectively equal to the identifiers of said list.
26. A method according to any one of the preceding claims, wherein a plurality of formats are provided for representing the data graph identifier lists,
15 and wherein each thesaurus entry contains an indication of the format used for representing the data graph identifier list associated therewith.
27. A method according to any one of the preceding claims, wherein said data representing data graph identifier lists comprise, for at least one thesaurus entry, coding data obtained by a coding scheme having n successive
20 coding layers, n being a number at least equal to 1, each layer having a predetermined pattern for dividing a range covering integers of an input list of said layer into subsets, said identifier list being the input list of the first layer for said thesaurus entry, wherein for any layer other than the last layer, an integer list representing the position, in the pattern of said layer, of each subset
25 containing at least one integer of the input list forms the input list for the next layer, and wherein said coding data comprise, for each layer and each subset containing at least one integer of the input list, data representing the position of each integer of the input list within said subset and, at least if said layer is the last layer, data representing the position of said subset in the pattern of said
30 layer.

WO 02/44943

PCT/IB01/02792

- 91 -

28. A method according to claim 27, wherein the coding data are stored in a plurality of files including files respectively allocated to thesaurus entries.
29. A method according to claim 27 or 28, wherein the coding data are stored in a plurality of files including at least one file allocated to a respective thesaurus, for containing the coding data relating to the entries of said thesaurus.
30. A method according to any one of claims 27 to 29, wherein the coding data are stored in at least one file allocated to a plurality of thesauruses, for containing the coding data relating to the entries of said plurality of thesauruses.
31. A method according to any one of claims 27 to 30, wherein the pattern of each layer is such that the integer subsets are consecutive intervals consisting of the same number of integers.
32. A method according to claim 31, wherein said number of integers is a whole power of 2 for each layer.
33. A method according to any one of claims 27 to 32, wherein the data representing the position of each integer of the input list of each layer within a subset consist of a bitmap segment in which each bit is associated with a respective integer of the subset to indicate whether said integer belongs to the input list of said layer.
34. A method according to any one of claims 27 to 33, wherein the position of each subset in the layer n pattern is represented by an integer rank which is included in the coding data, in association with the data representing the position of the integers of the input list of layer n within said subset.
35. A method according to any one of claims 27 to 33, wherein the position of a subset in the pattern of each layer is represented by an integer rank which is included in the coding data for said layer, in association with the data representing the position of the integers of the input list of said layer within said subset.

WO 02/44943

PCT/IB01/02792

- 92 -

36. A method according to any one of claims 27 to 35, wherein $n \geq 2$ and layer k data containers each having a plurality of records are provided in a computer memory for $1 \leq k \leq n$, each record of a layer k data container being associated with a layer k integer rank representing the position of a subset in the layer k pattern, and wherein each record of a layer k data container associated with a layer k rank representing the position of a subset in the layer k pattern has a first field for containing data for retrieving the position within said subset of any integer of a layer k input list relating to a data graph identifier list, whereby a combination of said layer k rank with any position retrievable from the data contained in said first field determines a layer $k-1$ rank with which a respective record of the layer $k-1$ data container is associated if $k > 1$, and an identifier of said data graph identifier list if $k = 1$.

37. A method according to claim 36, wherein, for $1 \leq k \leq n$, said data contained in the first field of a record of the layer k data container for retrieving the position of any integer of a layer k input list within a subset comprise a bitmap segment in which each bit is associated with a respective integer of said subset to indicate whether said integer belongs to said layer k input list.

38. A method according to claim 37, wherein, for $1 \leq k \leq n$, each record of the layer k data container associated with a layer k rank further has a second field for containing said layer k rank.

39. A method according to claim 38, wherein each data container comprises at least two files where the first and second fields of the records of said data container are respectively stored, said files being accessible separately.

40. A method according to claim 36, wherein, for $1 \leq k < n$, each record of the layer k data container further has a second field for containing a number representing the position of an integer of a layer $k+1$ input list within a subset of the layer $k+1$ pattern,

and wherein, for $1 < k \leq n$, said data contained in the first field of a record of the layer k data container associated with a layer k rank for retrieving

WO 02/44943

PCT/IB01/02792

- 93 -

the position of any integer of a layer k input list within a subset of the layer k pattern comprise a pointer to at least one record of the layer k-1 data container in which the second field contains a number representing the position of an integer of said layer k input list within said subset of the layer k pattern, 5 whereby said record of the layer k-1 data container is associated with the layer k-1 rank determined by the combination of said layer k rank with the position represented by said number.

41. A method according to claim 40, wherein said data contained in the first field of a record of the layer 1 data container for retrieving the position of 10 any integer of a data graph identifier list within a subset comprise a bitmap segment in which each bit is associated with a respective integer of said subset to indicate whether said integer represents a data graph identifier of said list.

42. A method according to claim 40 or 41, wherein each record of the layer n data container associated with a layer n rank further has a second field 15 for containing said layer n rank.

43. A method according to any one of claims 40 to 42, wherein each layer k data container for $1 \leq k < n$ comprises at least two files where the first and second fields of the records of said data container are respectively stored, said files being accessible separately.

20 44. A method according to any one of claims 36 to 43, wherein, for $1 \leq k \leq n$, each record of the layer k data container further has a next address field, whereby record chains are defined in the layer k data container by means of the next address fields, and wherein at least some of the thesaurus entries are respectively associated with record chains in the layer n data container, 25 whereby the coding data relating to one of said entries for layer n are stored in or retrievable from the record chain associated therewith in the layer n data container.

45. A method according to claim 44, wherein, for $1 \leq k < n$, said thesaurus entries are respectively associated with record chains in the layer k 30 data container, whereby the coding data relating to one of said entries for layer

WO 02/44943

PCT/IB01/02792

- 94 -

k are stored in or retrievable from the record chain associated therewith in the layer k data container.

46. A method according to claim 44, wherein, for $1 < k \leq n$, each record of the layer k data container further has a head address field for pointing to an address of a first record of a respective chain in the layer k-1 data container.

47. A method according to any one of claims 44 to 46, wherein each layer k data container for $1 \leq k \leq n$ comprises at least two files where the first fields and the next address fields of the records of said data container are respectively stored, said files being accessible separately.

48. A method according to any one of the preceding claims, wherein an integer range covering the identifiers allocated to the data graphs is partitioned into a plurality of predetermined portions, wherein at least some of the data representing identifier lists are distributed into a plurality of storage sections respectively associated with said portions, wherein a storage section associated with one of said portions contains data representing identifier sub-lists consisting of identifiers of said portion.

49. A method according to claim 48, wherein a respective storage unit is provided for each of said portions of the data graph identifier range, to receive the storage sections associated with said portion.

50. A method according to claim 49, wherein at least some of the word thesauruses have a plurality of sections respectively associated with said portions, wherein a section, associated with one of said portions, of a word thesaurus associated with an attribute has a respective entry for each word assigned to said attribute in a data graph to which an identifier of said portion is allocated, said entry containing data for retrieving an identifier sub-list from the storage section associated with said portion.

51. A method according to both claims 12 and 49, wherein at least some of the macroword thesauruses have a plurality of sections respectively associated with said portions, wherein a section, associated with one of said

WO 02/44943

PCT/IB01/02792

- 95 -

portions, of a macroword thesaurus associated with an attribute and with a prefix length has a respective entry for each prefix value having said prefix length and matching a corresponding prefix of at least one word assigned to said attribute in a data graph to which an identifier of said portion is allocated, said entry containing data for retrieving an identifier sub-list from the storage section associated with said portion.

52. A method according to claim 48 or 49, wherein each thesaurus entry has a plurality of fields respectively associated with said portions, for containing data for retrieving respective identifier sub-lists from the storage sections.

53. A method of processing a query in a database system, wherein a group of attributes is defined and words of a collection of data are assigned to said attributes, the group of attributes being divided into a plurality of sub-groups respectively associated with a plurality of data tables having independent numbers of records, with links between respective records from the data tables, wherein identifiers are respectively allocated to data graphs, each data graph representing related attribute values respectively assigned to the attributes of said group, each attribute value of a data graph being either a default value or a word of said collection of data, wherein a plurality of thesauruses each associated with a respective attribute of said group and data representing data graph identifier lists respectively associated with entries of said thesauruses are stored, wherein each thesaurus associated with one attribute is defined with reference to a partition into subsets of a set of words which can be assigned to said one attribute and has a respective entry for each subset including at least one word assigned to said one attribute in the collection of data, the data graph identifier list associated with said thesaurus entry including any identifier allocated to a data graph having a word of said subset assigned to said one attribute, the method comprising the steps of:

- analyzing query criteria to determine a combination involving thesaurus entries relevant to the query ;
- determining a matching data graph identifier list based on said combination and on the stored data representing the data graph identifier lists associated with said relevant thesaurus entries;

WO 02/44943

PCT/IB01/02792

- 96 -

- processing said matching data graph identifier list to output a response.

54. A method according to claim 53, wherein the step of analyzing the query criteria comprises, for at least one attribute referred to in said criteria:

- selecting at least one range of words defined for said attribute in the query criteria; and
- mapping the words of the selected range which are assigned to said attribute in the collection of data with one or more subsets, the thesaurus entry for each of said one or more subset being retained as a relevant entry for the selected range,

and wherein the step of determining the matching data graph identifier list comprises merging respective portions of the identifier lists represented by the data of the relevant thesaurus entries retained for said selected range.

55. A method according to claim 54, wherein the mapping is performed so as to retain a minimum number of relevant thesaurus entries for each selected range.

56. A method according to claim 54 or 55, wherein each thesaurus associated with an attribute is defined with reference to a partition such that each subset consists of one word or of consecutive words of the set of words which can be assigned to said attribute, the entries of said thesaurus being sorted based on the words assigned to said attribute, and wherein the step of analyzing the query criteria comprises at least one dichotomy search in at least one thesaurus for identifying relevant thesaurus entries.

57. A method according to claim 56, wherein the thesauruses comprise word thesauruses each associated with a respective attribute of the group, with reference to a partition into subsets each consisting of one word.

58. A method according to claim 57, wherein each word thesaurus associated with an attribute of the group to which the default value is assigned in at least one of the data graphs further has an entry for the default value, whereby one of said data graph identifier lists is associated with said thesaurus

WO 02/44943

PCT/IB01/02792

- 97 -

entry for the default value and includes any identifier allocated to a data graph having said default value assigned to said attribute.

59. A method according to claim 57 or 58, wherein the thesauruses further comprise at least one macroword thesaurus associated with an attribute of said group and with a prefix length shorter than a word length of said attribute, whereby said macroword thesaurus is defined with reference to a partition into subsets each consisting of words beginning by a common prefix having said prefix length.

60. A method according to claim 59, wherein at least one attribute of said group has a plurality of macroword thesauruses, associated with different prefix lengths.

61. A method according to any one of claims 54 to 60, wherein the step of analyzing the query criteria comprises determining said combination involving relevant thesaurus entries as a tree having at least one leaf node, each leaf node corresponding to at least one relevant thesaurus entry retained for a respective attribute.

62. A method according to claim 61, wherein said tree has a plurality of nodes including said at least one leaf node and at least one operator node, each operator node representing a Boolean operator applied to at least one partial criterion represented by another node of said tree, one of the operator nodes being a root node representing all the query criteria.

63. A method according to claim 62, wherein the nodes of said tree further include at least one preset node for which a data graph identifier list has been determined prior to said step of analyzing the query criteria.

64. A method according to claim 63, wherein the data graph identifier list of said preset node is determined from at least one matching data graph identifier list obtained when processing a previous query.

65. A method according to any one of claims 62 to 64, wherein the step of determining the matching data graph identifier list comprises obtaining a

WO 02/44943

PCT/IB01/02792

- 98 -

respective identifier list for each node of said tree, whereby the identifier list obtained for each leaf node corresponding to at least one relevant thesaurus entry is the merger of respective portions of the identifier lists associated with said at least one relevant thesaurus entry, and the identifier list obtained for
5 each operator node representing a Boolean operator applied to at least one partial criterion is obtained by applying said Boolean operator to the identifier lists obtained for the node representing said at least one partial criterion, said matching data graph identifier list being determined as the identifier list obtained for the root node.

10 66. A method according to claim 65, wherein each of said obtained identifier lists is produced in the form of a bitmap vector consisting of bits assigned to respective data graphs to indicate whether the identifiers allocated to said data graphs belong to said obtained list.

67. A method according to claim 65 or 66, wherein a coding scheme
15 comprising n successive coding layers is used to provide coding data representing the identifier list associated with a thesaurus entry, n being a number at least equal to 1, each layer having a predetermined pattern for dividing a range covering integers of an input list of said layer into subsets, said identifier list being the input list of the first layer for said thesaurus entry,
20 wherein for any layer other than the last layer, an integer list representing the position, in the pattern of said layer, of each subset containing at least one integer of the input list forms the input list for the next layer,

and wherein the coding data comprise, for each layer and each subset containing at least one integer of the input list, data representing the
25 position of each integer of the input list within said subset and, at least if said layer is the last layer, data representing the position of said subset in the pattern of said layer.

68. A method according to claim 67, wherein the pattern of each layer is
30 such that the integer subsets are consecutive intervals consisting of the same number of integers.

WO 02/44943

PCT/IB01/02792

- 99 -

69. A method according to claim 68, wherein said number of integers is a whole power of 2 for each layer.

70. A method according to any one of claims 67 to 69, wherein said data representing the position of an integer of an input list within a subset consist of
5 a bitmap segment.

71. A method according to any one of claims 67 to 70, wherein the step of determining the matching data graph identifier list comprises determining a layer n integer list for each node of said tree, whereby the layer n integer list determined for a leaf node consists of a layer n input list associated, in the
10 coding scheme, with the merger of the identifier lists represented in the relevant thesaurus entries to which said leaf node corresponds, and whereby the layer n integer list obtained for each operator node representing a Boolean operator applied to at least one partial criterion is obtained by applying said Boolean operator to the layer n integer lists determined for the nodes representing said
15 at least one partial criterion, and wherein a layer n result list is determined as the layer n integer list obtained for the root node.

72. A method according to claim 71, wherein the nodes of said tree further include at least one preset node for which a data graph identifier list has been determined prior to said step of analyzing the query criteria, said data
20 graph identifier list being subjected to the coding scheme to provide a layer n input list which is determined as said layer n integer list for said preset node.

73. A method according to claim 71 or 72, wherein, in the coding scheme, the coding data representing the position of each integer of an input list within a subset for the coding layer n define a layer n bitmap segment in
25 which each bit is associated with a respective integer of the subset to indicate whether said integer belongs to said input list, while the data representing the position of said subset in the layer n pattern comprise a layer n integer rank associated with said layer n bitmap segment, and wherein the step of determining a layer n integer list for a leaf node comprises:

30 - initializing a layer n bitmap vector with logical zeroes;

WO 02/44943

PCT/IB01/02792

- 100 -

- obtaining the layer n ranks and associated bitmap segments from the coding data for each relevant thesaurus entry to which said leaf node corresponds; and
 - for each of said layer n ranks, superimposing the layer n bitmap segment associated therewith onto a segment of said layer n bitmap vector having a position determined by said layer n rank, the superimposition being performed according to a bitwise Boolean OR operation,
- 5 said layer n list for the leaf node corresponding to the resulting layer n bitmap vector.
- 10 74. A method according to any one of claims 71 to 73, wherein $n > 1$ and the step of determining the matching data graph identifier list further comprises, for k decreasing from $n-1$ to 1, determining a layer k integer list for each node of said tree, whereby the layer k integer list determined for a leaf node consists of any integer of a layer k input list, associated in the coding
- 15 scheme with the identifier list represented in a relevant thesaurus entry to which said leaf node corresponds, which belongs to a layer k subset whose position is represented in the layer $k+1$ result list, and whereby the layer k integer list obtained for each operator node representing a Boolean operator applied to at least one partial criterion is obtained by applying said Boolean
- 20 operator to the layer k integer lists determined for the nodes representing said at least one partial criterion, wherein a layer k result list is determined as the layer k integer list obtained for the root node, and wherein said matching data graph identifier list corresponds to the determined layer 1 result list.
75. A method according to claim 74, wherein the nodes of said tree
- 25 further include at least one preset node for which a data graph identifier list has been determined prior to said step of analyzing the query criteria, said data graph identifier list being subjected to the coding scheme to provide a layer k input list which is determined as said layer k integer list for said preset node.
76. A method according to claim 74 or 75, wherein, in the coding
- 30 scheme, the coding data representing the position of each integer of an input list within a subset for a coding layer $k < n$ define a layer k bitmap segment in

WO 02/44943

PCT/IB01/02792

- 101 -

which each bit is associated with a respective integer of the subset to indicate whether said integer belongs to said input list, while the coding data further comprise a layer k integer rank associated with said layer k bitmap segment to represent the position of said subset in the layer k pattern, and wherein the

5 step of determining a layer k integer list for a leaf node comprises:

- initializing a layer k bitmap vector with logical zeroes;
- obtaining the layer k ranks from the coding data for each relevant thesaurus entry to which said leaf node corresponds; and
- selecting any obtained layer k rank belonging to the layer k+1 result list

10 and superimposing the associated layer k bitmap segment onto a segment of said layer k bitmap vector having a position determined by the selected layer k rank, the superimposition being performed according to a bitwise Boolean OR operation,

said layer k list for the leaf node corresponding to the resulting layer k bitmap

15 vector.

77. A method according to claim 76, wherein, for $1 \leq k < n$, the layer k ranks and the layer k bitmap segments associated therewith for at least one thesaurus entry are stored at corresponding addresses in distinct first and second files, and wherein the step of determining a layer k integer list for a leaf

20 node comprises:

- providing a rank table in a RAM memory, having records associated with the addresses in said first and second files;
- filling the rank table by writing any selected layer k rank into the rank table record associated with the address of the selected layer k rank in

25 said first file; and

- for any record of the filled rank table containing a layer k rank and associated with an address in the second file, reading the associated layer k bitmap segment at said address in the second file and superimposing the read layer k bitmap segment onto a segment of said

30 layer k bitmap vector having a position determined by said layer k rank.

WO 02/44943

PCT/IB01/02792

- 102 -

78. A method according to claim 74 or 75, wherein the step of determining the matching data graph identifier list further comprises, for any coding layer k such that $1 < k \leq n$, determining a layer k' filtering list for $k \leq k' \leq n$ consisting of the layer k' input list obtained by providing the layer k result list as an input list in layer k of the coding scheme,
- 5 wherein, in the coding scheme, the coding data representing the position of each integer of an input list within a subset for a coding layer $k < n$ define a layer k bitmap segment in which each bit is associated with a respective integer of the subset to indicate whether said integer belongs to said input list, while a layer k integer rank associated with said layer k bitmap segment represents the position of said subset in the layer k pattern, and wherein the step of determining a layer k integer list for a leaf node for $k < n$ comprises:
- /a/ initializing a layer k bitmap vector with logical zeroes;
 - 15 /b/ selecting the layer n ranks obtained from the coding data for each relevant thesaurus entry to which said leaf node corresponds, and setting $k' = n$;
 - /c/ for each selected layer k' rank:
 - /c1/ if the selected layer k' rank represents the position in the layer k' pattern of a subset which includes at least one integer of the layer k' filtering list, obtaining the layer k' bitmap segment with which the selected layer k' rank is associated;
 - /c2/ for any integer of the layer k' filtering list whose position within said subset is represented in said layer k' bitmap segment,
 - 20 selecting a respective layer $k'-1$ rank determined from the selected layer k' rank and said position represented in said layer k' bitmap segment;
 - 25 /c3/ if $k' > k+1$, executing step /c/ with k' decremented by one unit; and
 - 30 /c4/ if $k'-1 = k$, obtaining any layer k bitmap segment with which a selected layer $k'-1$ rank is associated, and superimposing said layer k bitmap segment onto a segment of said layer k bitmap vector having a position determined by said selected layer $k'-1$

WO 02/44943

PCT/IB01/02792

- 103 -

rank, the superimposition being performed according to a bitwise Boolean OR operation, said layer k list for the leaf node corresponding to the resulting layer k bitmap vector.

- 5 79. A method according to claim 78, wherein, for $1 \leq k < n$, the layer k bitmap segments for at least one thesaurus entry are stored in at least one layer k file at addresses respectively corresponding to the layer k ranks associated therewith, and wherein, for $1 \leq k < n$, the step of determining a layer k integer list for a leaf node comprises:
- 10 - providing a rank table in a RAM memory, having records associated with the addresses in said layer k file;
- filling the rank table by writing any selected layer k rank into the rank table record associated with the address corresponding to the selected layer k rank; and
- 15 - for any record of the filled rank table containing a layer k rank and associated with an address in said layer k file, reading the associated layer k bitmap segment at said address and superimposing the read layer k bitmap segment onto a segment of said layer k bitmap vector having a position determined by said layer k rank.
- 20 80. A method according to any one of claims 53 to 79, wherein a link table is stored, having a plurality of rows respectively associated with the data graphs and a plurality of columns respectively associated with the attribute sub-groups, wherein each row of the link table contains, in each one of the columns, either a value indicating that each attribute value represented in the data graph associated with said row and assigned to an attribute of the sub-group associated with said one of the columns is a default value or a link value for retrieving at least one stored word of the collection of data represented in the data graph associated with said row and assigned to an attribute of the sub-group associated with said one of the columns, and wherein the step of
- 25 processing the matching data graph identifier list comprises reading at least
- 30

WO 02/44943

PCT/IB01/02792

- 104 -

one value in any row of the link table associated with a data graph identified in the matching data graph identifier list.

81. A method according to claim 80, wherein said data tables are stored, wherein each link value contained in the column of the link table associated with an attribute sub-group comprises data for identifying a record of the data table associated with said sub-group, and wherein the step of processing the matching data graph identifier list further comprises reading at least part of any data table record identified by a link value read in a row of the link table.
82. A method according to any one of claims 53 to 79, wherein the step of processing the matching data graph identifier list comprises, for at least one attribute specified in the query, selecting a thesaurus associated with said attribute and detecting entries of the selected thesaurus with which identifier lists having a non-empty intersection with the matching data graph identifier list are associated.
83. A method according to claim 82, wherein said attribute specified in the query has $Q+1$ stored thesauruses associated with different prefix lengths, Q being an integer at least equal to 0, each of said $Q+1$ thesauruses having a thesaurus level parameter q such that $0 \leq q \leq Q$, whereby the prefix length is a decreasing function of the level parameter q and corresponds to a word length of said attribute for $q = 0$, wherein each of said $Q+1$ thesauruses is defined with reference to a respective partition into subsets each consisting of words beginning by a common prefix having the prefix length associated with said thesaurus, the entries of said thesaurus being sorted based on the prefix values.
84. A method according to claim 83, wherein, the selected thesaurus having a level parameter $QA \geq 0$, the detection of entries in the selected thesaurus comprises the steps of:
- /a/ providing respective level q target lists and respective level q thesaurus ranges covering consecutive entries of the level q thesaurus for $QA \leq q \leq Q$;

WO 02/44943

PCT/IB01/02792

- 105 -

- /b/ initializing the level Q target list with the matching data graph identifier list, initializing the level parameter q with the value Q, and selecting a first entry of the level Q thesaurus range;
 - /c/ determining an intersection list between the level q target list and the identifier list associated with the selected entry of the level q thesaurus range;
 - /d/ if the intersection list determined in the preceding step /c/ is empty, selecting another entry of the level q thesaurus range and repeating step /c/;
 - /e/ if q is greater than QA:
 - /e1/ setting the level q-1 target list as equal to the intersection list determined in the preceding step /c/;
 - /e2/ setting the level q-1 thesaurus range as consisting of the entries of the level q-1 thesaurus relating to level q-1 prefixes which begin with the level q prefix of the selected level q thesaurus entry, and selecting a first entry of the level q-1 thesaurus range;
 - /e3/ decrementing q by one unit and returning to step /c/;
 - /f/ if q is equal to QA:
 - /f1/ including the selected level QA thesaurus entry in the detected entries;
 - /f2/ if the level Q target list is equal to the intersection list determined in the preceding step /c/, terminating the detection of entries in the selected thesaurus;
 - /f3/ removing the integers of the intersection list determined in the preceding step /c/ from any target list including at least one integer which is not in said intersection list;
 - /f4/ setting q as the smallest level parameter for which the target list includes at least one integer which is not in said intersection list;
 - /f5/ selecting another entry in the level q thesaurus range and returning to step /c/.

WO 02/44943

PCT/IB01/02792

- 106 -

85. A method according to claim 84, wherein $Q \geq 1$ and each thesaurus having a level parameter $q \geq 1$ further contains, in each entry provided for a level q prefix value, data designating the entry of the level $q-1$ thesaurus which corresponds to the lowest or highest level $q-1$ prefix beginning with the level q prefix of said level q thesaurus entry, and wherein step /e2/ comprises selecting the level $q-1$ thesaurus entry designated in the selected level q thesaurus entry.

86. A method according to claim 84 or 85, wherein a coding scheme comprising n successive coding layers is used to provide coding data representing the identifier list associated with a level q thesaurus entry for $0 \leq q \leq Q$, n being a number at least equal to 1, each layer having a predetermined pattern for dividing a range covering integers of an input list of said layer into subsets, said identifier list being the input list of the first layer for said thesaurus entry, wherein for any layer other than the last layer, an integer list representing the position, in the pattern of said layer, of each subset containing at least one integer of the input list forms the input list for the next layer,

wherein the coding data comprise, for each layer and each subset containing at least one integer of the input list, data representing the position of each integer of the input list within said subset and, at least if said layer is the last layer, data representing the position of said subset in the pattern of said layer,

and wherein each level q target list forms a layer 1 and level q filtering list and is submitted as a layer 1 input list in the coding scheme for $QA \leq q \leq Q$ to provide respective layer k and level q filtering lists for $1 < k \leq n$ if $n > 1$, said layer k and level q filtering list provided from a level q target list being the layer k input list obtained from said level q target list in the coding scheme.

87. A method according to claim 86, wherein the pattern of each layer is such that the integer subsets are consecutive intervals consisting of the same number of integers.

WO 02/44943

PCT/IB01/02792

- 107 -

88. A method according to claim 87, wherein said number of integers is a whole power of 2 for each layer.
89. A method according to any one of claims 86 to 88, wherein the step /c/ of determining the intersection list between a level q target list and an identifier list comprises, from $k = n$:
- /c1/ computing a layer k intersection list between the layer k input list obtained from said identifier list in the coding scheme and the layer k and level q filtering list corresponding to said level q target list;
 - /c2/ if the computed layer k intersection list is empty, determining said intersection list between the level q target list and the identifier list as being empty;
 - /c3/ if $k = 1$, determining said intersection list between the level q target list and the identifier list as the computed layer 1 intersection list; and
 - /c4/ if $k > 1$, decrementing k by one unit and repeating from step /c1/.
90. A method according to claim 89, wherein, in the coding scheme, the coding data representing the position of each integer of an input list within a subset for a coding layer $k \leq n$ define a layer k bitmap segment in which each bit is associated with a respective integer of the subset to indicate whether said integer belongs to said input list, while the data representing the position of said subset in the layer k pattern comprise a layer k integer rank associated with said layer k bitmap segment, and wherein the step /c1/ of computing a layer k intersection list between a layer k input list obtained from an identifier list in the coding scheme and a layer k and level q filtering list, represented by a first layer k bitmap vector, comprises:
- initializing a second layer k bitmap vector with logical zeroes;
 - obtaining layer k ranks from the coding data representing said identifier list; and
 - selecting any obtained layer k rank which represents the position in the layer k pattern of a subset including at least one integer of said layer k and level q filtering list, obtaining the layer k bitmap segment with which the selected layer k rank is associated, and determining a segment of

WO 02/44943

PCT/IB01/02792

- 108 -

the second layer k bitmap vector having a position determined by the selected layer k rank by combining the obtained layer k bitmap segment with a segment of the first layer k bitmap vector having a position determined by the selected layer k rank according to a bitwise Boolean AND operation,

5 said layer k intersection list corresponding to the resulting second layer k bitmap vector.

91. A method according to claim 90, wherein, for $1 \leq k < n$, the layer k ranks and the layer k bitmap segments associated therewith for at least one thesaurus entry are stored at corresponding addresses in distinct first and second files, and wherein the step /c1/ of computing a layer k intersection list between a layer k input list obtained from an identifier list in the coding scheme and a layer k and level q filtering list, represented by a first layer k bitmap vector, comprises:

15 - providing a rank table in a RAM memory, having records associated with the addresses in said first and second files;

- filling the rank table by writing any selected layer k rank into the rank table record associated with the address of said selected layer k rank in said first file; and

20 - for any record of the filled rank table containing a layer k rank and associated with an address in the second file, reading the associated layer k bitmap segment at said address in the second file and combining the read layer k bitmap segment with a segment of the first layer k bitmap vector having a position determined by said layer k rank

25 according to a bitwise Boolean AND operation to determine a segment of the second layer k bitmap vector having a position determined by said layer k rank.

92. A method according to any one of claims 86 to 88, wherein $n > 1$ and in the coding scheme, the coding data representing the position of each integer of an input list within a subset for a coding layer $k \leq n$ define a layer k bitmap segment in which each bit is associated with a respective integer of the

30

WO 02/44943

PCT/IB01/02792

- 109 -

subset to indicate whether said integer belongs to said input list, while a layer k integer rank associated with said layer k bitmap segment represents the position of said subset in the layer k pattern, and wherein the step /c/ of determining the intersection list between a level q target list, corresponding to layer k and level q filtering lists represented by a respective first layer k bitmap vectors for $1 \leq k \leq n$, and an identifier list comprises:

- /c1/ initializing a second bitmap vector with logical zeroes;
- /c2/ selecting layer n ranks obtained from the coding data representing said identifier list, and setting $k = n$;
- 10 /c3/ for each selected layer k rank:
 - /c31/ if the selected layer k rank represents the position in the layer k pattern of a subset which includes at least one integer of said layer k and level q filtering list, obtaining the layer k bitmap segment with which the selected layer k rank is associated;
 - 15 /c32/ for any integer of the layer k and level q filtering list whose position within said subset is represented in said layer k bitmap segment, selecting a respective layer k-1 rank determined from the selected layer k rank and said position represented in said layer k bitmap segment;
 - 20 /c33/ if $k > 2$, executing step /c3/ with k decremented by one unit; and
 - /c34/ if $k = 2$, obtaining any layer 1 bitmap segment with which a selected layer 1 rank is associated, and combining the
 - 25 obtained layer 1 bitmap segment with a segment of the first layer 1 bitmap vector having a position determined by said layer 1 rank according to a bitwise Boolean AND operation to determine a segment of the second bitmap vector having a position determined by said layer 1 rank,
 - 30 said intersection list corresponding to the resulting second bitmap vector.

93. A method according to claim 92, wherein the layer 1 bitmap segments for at least one thesaurus entry are stored in at least one layer 1 file

WO 02/44943

PCT/IB01/02792

- 110 -

at addresses respectively corresponding to the layer 1 ranks associated therewith, and the step /c/ of determining an intersection list comprises:

- providing a rank table in a RAM memory, having records associated with the addresses in said layer 1 file;
- 5 - filling the rank table by writing any layer 1 rank selected in step /c32/ into the rank table record associated with the address corresponding to the selected layer 1 rank; and
- for any record of the filled rank table containing a layer 1 rank and associated with an address in said layer 1 file, reading the associated
- 10 layer 1 bitmap segment at said address and combining the read layer 1 bitmap segment with a segment of the first layer 1 bitmap vector having a position determined by said layer 1 rank according to a bitwise Boolean AND operation to determine a segment of the second layer 1 bitmap vector having a position determined by said layer 1 rank.

- 15 94. A method according to any one of claims 90 to 93, further comprising determining a pre-filtering flag for each entry of a level q thesaurus, said pre-filtering flag having a first value when said entry is associated with a data graph identifier list represented by coding data which do not define any layer n rank representing the position in the layer n pattern of a subset which
- 20 includes at least one integer of a layer n and level q filtering list, and wherein the step /c/ of determining the intersection list between a level q target list, corresponding to said layer n and level q filtering list, and an identifier list associated with an entry of the level q thesaurus comprises determining said intersection list as being empty if the pre-filtering flag determined for said entry
- 25 has said first value.

95. A method according to claim 94, wherein, for any level q thesaurus entry associated with a data graph identifier list represented by coding data which define a layer n rank representing the position in the layer n pattern of a subset which includes at least one integer of the layer n and level q filtering list,
- 30 the layer n bitmap segment associated with said layer n rank is obtained and said first value is allocated to the pre-filtering flag determined for said entry if

WO 02/44943

PCT/IB01/02792

- 111 -

the obtained layer n bitmap segment does not represent the position of any integer of said layer n and level q filtering list within said subset.

96. A method according to any one of claims 82 to 95, wherein the step of processing the matching data graph identifier list further comprises writing
5 output data associated with any detected entry of a selected thesaurus into an output table.

97. A method according to claim 96, wherein the output table includes a respective row corresponding to each identifier of the matching data graph identifier list, and wherein output data associated with a detected entry of a
10 selected thesaurus are written into any row of the output table corresponding to a data graph identifier belonging to both the matching data graph identifier list and the identifier list associated with said detected thesaurus entry.

98. A method according to claim 97, wherein each data graph identifier has a respective row of the output table corresponding thereto, wherein the
15 rows of the output table are initialized with a default value before writing the output data, and wherein the rows of the output table which do not contain the default value are read after writing the output data.

99. A method according to claim 97, wherein the output table is associated with an index file having a respective record for each data graph
20 identifier, containing either a default value or a pointer designating a respective row of the output table corresponding to said data graph identifier, wherein the records of the index file are initialized with a default value before writing the output data, and wherein the step of writing output data associated with a detected entry of a first selected thesaurus comprises, for each data graph
25 identifier belonging to both the matching data graph identifier list and the identifier list associated with said detected entry of the first selected thesaurus:

- allocating an available row of the output table to correspond to said data graph identifier,
- writing output data into the allocated row; and

WO 02/44943

PCT/IB01/02792

- 112 -

- writing a pointer to the allocated row into the record of the index file provided for said data graph identifier.

100. A method according to any one of claims 97 to 99, wherein the output table has a plurality of columns each associated with a respective attribute for which a thesaurus is selected, and wherein output data associated with a detected entry of a thesaurus selected for an attribute associated with a column of the output table are written into said column.

101. A method according to both claims 99 and 100, wherein the step of writing output data associated with a detected entry of at least one second selected thesaurus comprises, for each data graph identifier belonging to both the matching data graph identifier list and the identifier list associated with said detected entry of the second selected thesaurus:

- reading the pointer contained in the record of the index file provided for said data graph identifier; and
- writing output data into the row of the output table designated by said pointer.

102. A method according to any one of claims 96 to 101, wherein the selected thesaurus being a word thesaurus defined with reference to a partition into subsets each consisting of one word, the output data associated with a detected entry comprise the word for which said detected entry is provided.

103. A method according to any one of claims 96 to 101, wherein the selected thesaurus being a macroword thesaurus associated with a prefix length and defined with reference to a partition into subsets each consisting of words beginning by a common prefix having said prefix length, the output data associated with a detected entry comprise the prefix value for which said detected entry is provided.

104. A method according to any one of claims 96 to 101, wherein the output data associated with a detected entry comprise an address of said detected entry in the selected thesaurus.

WO 02/44943

PCT/IB01/02792

- 113 -

105. A method according to any one of claims 96 to 101, wherein the output data associated with a detected entry of a selected thesaurus comprise a numerical value derived from said thesaurus entry.

106. A method according to claim 105, wherein, for a detected entry of at least one selected thesaurus, said numerical value is calculated by applying a mathematical function to a thesaurus value stored in said entry.

107. A method according to claim 105 or 106, wherein, for a detected entry of at least one selected thesaurus, said numerical value is calculated by applying a mathematical function to a plurality of values including a thesaurus value stored in said entry and at least one value already present in the output table.

108. A method according to any one of claims 105 to 107, wherein the output table includes a respective row corresponding to each identifier of the matching data graph identifier list, and wherein a numerical value derived from a detected thesaurus entry is written into any row of the output table corresponding to a data graph identifier belonging to both the matching data graph identifier list and the identifier list associated with said detected thesaurus entry.

109. A method according to claim 108, wherein the numerical value, derived from a detected entry of a first selected thesaurus and written into any row of the output table corresponding to a data graph identifier belonging to both the matching data graph identifier list and the identifier list associated with said entry of the first selected thesaurus, is obtained from a thesaurus value stored in said entry,

and wherein the numerical value, derived from a detected entry of at least one second selected thesaurus and written into a row of the output table corresponding to a data graph identifier belonging to both the matching data graph identifier list and the identifier list associated with said entry of the second selected thesaurus, is calculated by applying a mathematical function

WO 02/44943

PCT/IB01/02792

- 114 -

to a plurality of values including a thesaurus value stored in said entry and at least one value already present in said row of the output table.

110. A method according to claim 108 or 109, further comprising calculating an output value from a set of numerical values which have been respectively written into the rows of the output table.

111. A method according to any one of claims 53 to 110, wherein an integer range covering the identifiers allocated to the data graphs is partitioned into a plurality of predetermined portions, wherein at least some of the data representing identifier lists are distributed into a plurality of storage sections respectively associated with said portions, wherein a storage section associated with one of said portions contains data representing identifier sub-lists consisting of identifiers of said portion,

and wherein the step of determining a matching data graph identifier list is executed separately for the different portions of the data graph identifier range, by means of the respective storage sections.

112. A method according to claim 111, wherein the step of processing the matching data graph identifier is at least partially executed separately for the different portions of the data graph identifier range, by means of the respective storage sections.

113. A method according to claim 111 or 112, wherein the thesauruses have a plurality of sections respectively associated with said portions, wherein a section, associated with one of said portions, of a thesaurus associated with an attribute and defined with reference to a partition into subsets has a respective entry for each subset of said partition which includes at least one word assigned to said attribute in a data graph to which an identifier of said portion is allocated, said entry containing data representing an identifier sub-list including each identifier of said portion allocated to a data graph having a word of said subset assigned to said attribute, and wherein the step of analyzing the query criteria is at least partially executed separately for the different portions of the data graph identifier range, by means of the respective thesaurus sections.

WO 02/44943

PCT/IB01/02792

- 115 -

114. A method according to any one of claims 111 to 113, wherein the separate step executions are carried out in parallel by respective processors for the different portions of the data graph identifier range.

115. A method according to claim 114, wherein each thesaurus entry has a plurality of fields respectively associated with said portions, for containing data for retrieving respective identifier sub-lists from the storage sections, wherein the step of analyzing the query criteria is executed centrally for all the portions of the data graph identifier range, and wherein the relevant thesaurus entries used by a processor executing the step of determining a matching data graph identifier list by means of a storage section are designated by the data for retrieving identifier sub-lists from said storage section.

116. A method according to claim 115, wherein the step of analyzing the query criteria is executed by a query server connected to said processors through a communication network.

117. A method according to any one of claims 114 to 116, wherein a list update server is connected, through the communication network, to a plurality of storage units respectively coupled to said processors, the list update server controlling the storage units to maintain the storage sections.

118. A method of organizing information in a database system, wherein a plurality of row identifiers are defined to designate respective rows of a reference table having columns respectively associated with data attributes, said rows containing groups of related words assigned to said attributes in a collection of data, the method comprising the steps of:

- storing at least one macroword thesaurus associated with one of the attributes and with a prefix length shorter than a word length of said attribute, said macroword thesaurus having a respective entry for each prefix value having said prefix length and matching a corresponding prefix of at least one word assigned to said data attribute in the collection of data; and

WO 02/44943

PCT/IB01/02792

- 116 -

- storing data representing identifier lists respectively associated with the macroword thesaurus entries, wherein the identifier list associated with an entry, relating to a prefix value, of a macroword thesaurus associated with an attribute includes any row identifier designating a row of the reference table having a word whose corresponding prefix matches said prefix value in the column associated with said attribute.

119. A method according to claim 118, wherein the entries of each macroword thesaurus associated with an attribute are sorted based on the prefix values.

120. A method according to claim 118 or 119, wherein a plurality of macroword thesauruses associated with different prefix lengths are stored for at least one attribute.

121. A method according to any one of claims 118 to 120, further comprising the step of storing a word thesaurus associated with said one of the attributes, said word thesaurus having a respective entry for each word assigned at least once to said attribute in the collection of data, said entry containing data representing an identifier list including each row identifier designating a row of the reference table having said word in the column associated with said attribute.

122. A method according to claim 121, wherein the word thesaurus associated with an attribute for which the reference table has a default value in at least one row further has an entry for the default value, containing data representing an identifier list including each row identifier designating a row of the reference table having said default value in the column associated with said attribute.

123. A method according to claim 121 or 122, wherein the entries of the word thesaurus are sorted based on the words assigned to said attribute.

124. A method according to both claims 119 and 123, wherein at least one attribute has a number Q of stored macroword thesauruses associated

WO 02/44943

PCT/IB01/02792

- 117 -

with different prefix lengths, each having a thesaurus level parameter q such that $1 \leq q \leq Q$, Q being an integer at least equal to 1, the prefix length being a decreasing function of the level parameter if $Q > 1$, wherein the level 1 macroword thesaurus further contains, in each entry provided for a level 1 prefix value, data designating the entry of the word thesaurus associated with said attribute which corresponds to the lowest or highest word whose corresponding prefix matches said level 1 prefix value, and wherein any macroword thesaurus having a level parameter $q > 1$ further contains, in each entry provided for a level q prefix value, data designating the entry of the level $q-1$ macroword thesaurus which corresponds to the lowest or highest level $q-1$ macroword whose corresponding prefix matches said level q prefix value.

125. A method according to any one of claims 118 to 124, wherein said reference table is a virtual table which is not stored.

126. A method according to claim 125, further comprising the step of storing a link table having a plurality of rows respectively associated with the rows of the reference table and a plurality of columns respectively associated with attribute sub-groups, wherein each row of the link table contains, in each one of the columns, either a value indicating that each attribute value represented in the associated reference table row and assigned to an attribute of the sub-group associated with said one of the columns is a default value or a link value for retrieving at least one stored word of the collection of data represented in the associated reference table row and assigned to an attribute of the sub-group associated with said one of the columns.

127. A method according to claim 126, wherein a respective data table is stored for each of the attribute sub-groups, and wherein each link value contained in a column of the link table associated with an attribute sub-group comprises data for identifying a row of the data table stored for said sub-group.

128. A method according to any one of claims 118 to 127, wherein said data representing identifier lists comprise, for at least one thesaurus entry, coding data obtained by a coding scheme having n successive coding layers, n

WO 02/44943

PCT/IB01/02792

- 118 -

being a number at least equal to 1, each layer having a predetermined pattern for dividing a range covering integers of an input list of said layer into subsets, said identifier list being the input list of the first layer for said thesaurus entry, wherein for any layer other than the last layer, an integer list representing the position, in the pattern of said layer, of each subset containing at least one integer of the input list forms the input list for the next layer, and wherein said coding data comprise, for each layer and each subset containing at least one integer of the input list, data representing the position of each integer of the input list within said subset and, at least if said layer is the last layer, data representing the position of said subset in the pattern of said layer.

129. A method according to claim 128, wherein the coding data are stored in a plurality of files including files respectively allocated to thesaurus entries.

130. A method according to claim 128 or 129, wherein the coding data are stored in a plurality of files including at least one file allocated to a respective thesaurus, for containing the coding data relating to the entries of said thesaurus.

131. A method according to any one of claims 128 to 130, wherein the coding data are stored in at least one file allocated to a plurality of thesauruses, for containing the coding data relating to the entries of said plurality of thesauruses.

132. A method according to any one of claims 128 to 131, wherein the pattern of each layer is such that the integer subsets are consecutive intervals consisting of the same number of integers.

133. A method according to claim 132, wherein said number of integers is a whole power of 2 for each layer.

134. A method according to any one of claims 128 to 133, wherein $n \geq 2$ and layer k data containers each having a plurality of records are provided in a computer memory for $1 \leq k \leq n$, each record of a layer k data container being

WO 02/44943

PCT/IB01/02792

- 119 -

- associated with a layer k integer rank representing the position of a subset in the layer k pattern, and wherein each record of a layer k data container associated with a layer k rank representing the position of a subset in the layer k pattern has a first field for containing data for retrieving the position within said subset of any integer of a layer k input list relating to an identifier list, whereby a combination of said layer k rank with any position retrievable from the data contained in said first field determines a layer k-1 rank with which a respective record of the layer k-1 data container is associated if $k > 1$, and an identifier of said identifier list if $k = 1$.
- 10 135. A method according to claim 134, wherein, for $1 \leq k \leq n$, said data contained in the first field of a record of the layer k data container for retrieving the position of any integer of a layer k input list within a subset comprise a bitmap segment in which each bit is associated with a respective integer of said subset to indicate whether said integer belongs to said layer k input list.
- 15 136. A method according to claim 135, wherein, for $1 \leq k \leq n$, each record of the layer k data container associated with a layer k rank further has a second field for containing said layer k rank.
137. A method according to claim 136, wherein each data container comprises at least two files where the first and second fields of the records of said data container are respectively stored, said files being accessible separately.
- 20 138. A method according to claim 134, wherein, for $1 \leq k < n$, each record of the layer k data container further has a second field for containing a number representing the position of an integer of a layer k+1 input list within a subset of the layer k+1 pattern,
- 25 and wherein, for $1 < k \leq n$, said data contained in the first field of a record of the layer k data container associated with a layer k rank for retrieving the position of any integer of a layer k input list within a subset of the layer k pattern comprise a pointer to at least one record of the layer k-1 data container
- 30 in which the second field contains a number representing the position of an

integer of said layer k input list within said subset of the layer k pattern, whereby said record of the layer k-1 data container is associated with the layer k-1 rank determined by the combination of said layer k rank with the position represented by said number.

- 5 139. A method according to claim 138, wherein said data contained in the first field of a record of the layer 1 data container for retrieving the position of any integer of an identifier list within a subset comprise a bitmap segment in which each bit is associated with a respective integer of said subset to indicate whether said integer represents an identifier of said list.
- 10 140. A method according to claim 138 or 139, wherein each record of the layer n data container associated with a layer n rank further has a second field for containing said layer n rank.
141. A method according to any one of claims 138 to 140, wherein each layer k data container for $1 \leq k < n$ comprises at least two files where the first and second fields of the records of said data container are respectively stored, 15 said files being accessible separately.
142. A method according to any one of claims 134 to 141, wherein, for $1 \leq k \leq n$, each record of the layer k data container further has a next address field, whereby record chains are defined in the layer k data container by means 20 of the next address fields, and wherein at least some of the thesaurus entries are respectively associated with record chains in the layer n data container, whereby the coding data relating to one of said entries for layer n are stored in or retrievable from the record chain associated therewith in the layer n data container.
- 25 143. A method according to claim 142, wherein, for $1 \leq k < n$, said thesaurus entries are respectively associated with record chains in the layer k data container, whereby the coding data relating to one of said entries for layer k are stored in or retrievable from the record chain associated therewith in the layer k data container.

WO 02/44943

PCT/IB01/02792

- 121 -

144. A method according to claim 142, wherein, for $1 < k \leq n$, each record of the layer k data container further has a head address field for pointing to an address of a first record of a respective chain in the layer k-1 data container.

145. A method according to any one of claims 142 to 144, wherein each layer k data container for $1 \leq k \leq n$ comprises at least two files where the first fields and the next address fields of the records of said data container are respectively stored, said files being accessible separately.

146. A method according to any one of claims 118 to 145, wherein an integer range covering the identifiers designating the rows of the reference table is partitioned into a plurality of predetermined portions, wherein at least some of the data representing identifier lists are distributed into a plurality of storage sections respectively associated with said portions, wherein a storage section associated with one of said portions contains data representing identifier sub-lists consisting of identifiers of said portion.

147. A method according to claim 146, wherein a respective storage unit is provided for each of said portions of the reference table row identifier range, to receive the storage sections associated with said portion.

148. A method according to claim 147, wherein at least some of the macroword thesauruses have a plurality of sections respectively associated with said portions, wherein a section, associated with one of said portions, of a macroword thesaurus associated with an attribute and with a prefix length has a respective entry for each prefix value having said prefix length and matching a corresponding prefix of at least one word assigned to said data attribute in a reference table row to which an identifier of said portion is allocated, said entry containing data for retrieving an identifier sub-list from the storage section associated with said portion.

149. A method according to both claims 121 and 148, wherein at least one word thesaurus has a plurality of sections respectively associated with said portions, wherein a section, associated with one of said portions, of a word thesaurus associated with an attribute has a respective entry for each word

WO 02/44943

PCT/IB01/02792

- 122 -

assigned to said attribute in a reference table row to which an identifier of said portion is allocated, said entry containing data for retrieving an identifier sub-list from the storage section associated with said portion.

150. A method according to claim 146 or 147, wherein each thesaurus entry has a plurality of fields respectively associated with said portions, for containing data for retrieving respective identifier sub-lists from the storage sections.

151. A method of processing a query in a database system, wherein a plurality of row identifiers are defined to designate respective rows of a reference table having columns respectively associated with data attributes, said rows containing groups of related words assigned to said attributes in a collection of data, wherein a plurality of thesauruses each associated with a respective attribute and data representing reference table row identifier lists respectively associated with entries of said thesauruses are stored, wherein each thesaurus associated with one attribute is defined with reference to a partition into subsets of a set of words which can be assigned to said one attribute and has a respective entry for each subset including at least one word assigned to said one attribute in the collection of data, the reference table row identifier list associated with said thesaurus entry including any identifier allocated to a row of the reference table having a word of said subset assigned to said one attribute,

wherein the thesaurus include at least one macroword thesaurus associated with an attribute and with a prefix length shorter than a word length of said attribute, whereby said macroword thesaurus is defined with reference to a partition into subsets each consisting of words beginning by a common prefix having said prefix length, the method comprising the steps of:

- analyzing query criteria to determine a combination involving thesaurus entries relevant to the query ;
- determining a matching reference table row identifier list based on said combination and on the stored data representing the reference table row identifier lists associated with said relevant thesaurus entries; and
- processing said matching row identifier list to output a response.

WO 02/44943

PCT/IB01/02792

- 123 -

152. A method according to claim 151, wherein at least one attribute has a plurality of macroword thesauruses, associated with different prefix lengths.

153. A method according to claim 151 or 152, wherein the step of analyzing the query criteria comprises, for at least one attribute referred to in said criteria:

- selecting at least one range of words defined for said attribute in the query criteria; and
 - mapping the words of the selected range which are assigned to said attribute in the collection of data with one or more subsets, the thesaurus entry for each of said one or more subset being retained as a relevant entry for the selected range,
- and wherein the step of determining the matching row identifier list comprises merging respective portions of the identifier lists represented by the data of the relevant thesaurus entries retained for said selected range.

154. A method according to claim 153, wherein the mapping is performed so as to retain a minimum number of relevant thesaurus entries for each selected range.

155. A method according to claim 153 or 154, wherein each thesaurus associated with an attribute is defined with reference to a partition such that each subset consists of one word or of consecutive words of the set of words which can be assigned to said attribute, the entries of said thesaurus being sorted based on the words assigned to said attribute, and wherein the step of analyzing the query criteria comprises at least one dichotomy search in at least one thesaurus for identifying relevant thesaurus entries.

156. A method according to claim 155, wherein the thesauruses further comprise at least one word thesaurus associated with a respective attribute, with reference to a partition into subsets each consisting of one word.

157. A method according to claim 156, wherein each word thesaurus associated with an attribute to which the default value is assigned in at least one of the reference table rows further has an entry for the default value,

whereby one of said reference table row identifier lists is associated with said thesaurus entry for the default value and includes any identifier allocated to a reference table row having said default value assigned to said attribute.

158. A method according to any one of claims 143 to 157, wherein the
5 step of analyzing the query criteria comprises determining said combination involving relevant thesaurus entries as a tree having at least one leaf node, each leaf node corresponding to at least one relevant thesaurus entry retained for a respective attribute.

159. A method according to claim 158, wherein said tree has a plurality of
10 nodes including said at least one leaf node and at least one operator node, each operator node representing a Boolean operator applied to at least one partial criterion represented by another node of said tree, one of the operator nodes being a root node representing all the query criteria.

160. A method according to claim 159, wherein the nodes of said tree
15 further include at least one preset node for which a reference table row identifier list has been determined prior to said step of analyzing the query criteria.

161. A method according to claim 160, wherein the reference table row
20 identifier list of said preset node is determined from at least one matching reference table row identifier list obtained when processing a previous query.

162. A method according to any one of claims 159 to 161, wherein the
step of determining the matching row identifier list comprises obtaining a
25 respective identifier list for each node of said tree, whereby the identifier list obtained for each leaf node corresponding to at least one relevant thesaurus entry is the merger of respective portions of the identifier lists associated with said at least one relevant thesaurus entry, and the identifier list obtained for each operator node representing a Boolean operator applied to at least one partial criterion is obtained by applying said Boolean operator to the identifier lists obtained for the node representing said at least one partial criterion, said

WO 02/44943

PCT/IB01/02792

- 125 -

matching row identifier list being determined as the identifier list obtained for the root node.

163. A method according to claim 162, wherein each of said obtained identifier lists is produced in the form of a bitmap vector consisting of bits
5 assigned to respective reference table rows to indicate whether the identifiers allocated to said reference table rows belong to said obtained list.

164. A method according to claim 162 or 163, wherein a coding scheme comprising n successive coding layers is used to provide coding data representing the identifier list associated with a thesaurus entry, n being a
10 number at least equal to 1, each layer having a predetermined pattern for dividing a range covering integers of an input list of said layer into subsets, said identifier list being the input list of the first layer for said thesaurus entry, wherein for any layer other than the last layer, an integer list representing the position, in the pattern of said layer, of each subset containing at least one
15 integer of the input list forms the input list for the next layer,

and wherein the coding data comprise, for each layer and each subset containing at least one integer of the input list, data representing the position of each integer of the input list within said subset and, at least if said layer is the last layer, data representing the position of said subset in the
20 pattern of said layer.

165. A method according to claim 164, wherein the pattern of each layer is such that the integer subsets are consecutive intervals consisting of the same number of integers.

166. A method according to claim 165, wherein said number of integers is
25 a whole power of 2 for each layer.

167. A method according to any one of claims 164 to 166, wherein said data representing the position of an integer of an input list within a subset consist of a bitmap segment.

WO 02/44943

PCT/IB01/02792

- 126 -

168. A method according to any one of claims 164 to 167, wherein the step of determining the matching row identifier list comprises determining a layer n integer list for each node of said tree, whereby the layer n integer list determined for a leaf node consists of a layer n input list associated, in the coding scheme, with the merger of the identifier lists represented in the relevant thesaurus entries to which said leaf node corresponds, and whereby the layer n integer list obtained for each operator node representing a Boolean operator applied to at least one partial criterion is obtained by applying said Boolean operator to the layer n integer lists determined for the nodes representing said at least one partial criterion, and wherein a layer n result list is determined as the layer n integer list obtained for the root node.

169. A method according to claim 168, wherein the nodes of said tree further include at least one preset node for which a reference table row identifier list has been determined prior to said step of analyzing the query criteria, said reference table row identifier list being subjected to the coding scheme to provide a layer n input list which is determined as said layer n integer list for said preset node.

170. A method according to claim 168, wherein, in the coding scheme, the coding data representing the position of each integer of an input list within a subset for the coding layer n define a layer n bitmap segment in which each bit is associated with a respective integer of the subset to indicate whether said integer belongs to said input list, while the data representing the position of said subset in the layer n pattern comprise a layer n integer rank associated with said layer n bitmap segment, and wherein the step of determining a layer n integer list for a leaf node comprises:

- initializing a layer n bitmap vector with logical zeroes;
- obtaining the layer n ranks and associated bitmap segments from the coding data for each relevant thesaurus entry to which said leaf node corresponds; and
- for each of said layer n ranks, superimposing the layer n bitmap segment associated therewith onto a segment of said layer n bitmap vector

WO 02/44943

PCT/IB01/02792

- 127 -

having a position determined by said layer n rank, the superimposition being performed according to a bitwise Boolean OR operation, said layer n list for the leaf node corresponding to the resulting layer n bitmap vector.

5 171. A method according to any one of claims 168 to 170, wherein $n > 1$ and the step of determining the matching row identifier list further comprises, for k decreasing from $n-1$ to 1, determining a layer k integer list for each node of said tree, whereby the layer k integer list determined for a leaf node consists of any integer of a layer k input list, associated in the coding scheme with the
10 identifier list represented in a relevant thesaurus entry to which said leaf node corresponds, which belongs to a layer k subset whose position is represented in the layer $k+1$ result list, and whereby the layer k integer list obtained for each operator node representing a Boolean operator applied to at least one partial criterion is obtained by applying said Boolean operator to the layer k integer
15 lists determined for the nodes representing said at least one partial criterion, wherein a layer k result list is determined as the layer k integer list obtained for the root node, and wherein said matching row identifier list corresponds to the determined layer 1 result list.

172. A method according to claim 171, wherein the nodes of said tree
20 further include at least one preset node for which a reference table row identifier list has been determined prior to said step of analyzing the query criteria, said reference table row identifier list being subjected to the coding scheme to provide a layer k input list which is determined as said layer k integer list for said preset node.

25 173. A method according to claim 171 or 172, wherein, in the coding scheme, the coding data representing the position of each integer of an input list within a subset for a coding layer $k < n$ define a layer k bitmap segment in which each bit is associated with a respective integer of the subset to indicate whether said integer belongs to said input list, while the coding data further
30 comprise a layer k integer rank associated with said layer k bitmap segment to

WO 02/44943

PCT/IB01/02792

- 128 -

represent the position of said subset in the layer k pattern, and wherein the step of determining a layer k integer list for a leaf node comprises:

- initializing a layer k bitmap vector with logical zeroes;
 - obtaining the layer k ranks from the coding data for each relevant thesaurus entry to which said leaf node corresponds; and
 - selecting any obtained layer k rank belonging to the layer k+1 result list and superimposing the associated layer k bitmap segment onto a segment of said layer k bitmap vector having a position determined by the selected layer k rank, the superimposition being performed according to a bitwise Boolean OR operation,
- said layer k list for the leaf node corresponding to the resulting layer k bitmap vector.

174. A method according to claim 173, wherein, for $1 \leq k < n$, the layer k ranks and the layer k bitmap segments associated therewith for at least one thesaurus entry are stored at corresponding addresses in distinct first and second files, and wherein the step of determining a layer k integer list for a leaf node comprises:

- providing a rank table in a RAM memory, having records associated with the addresses in said first and second files;
- filling the rank table by writing any selected layer k rank into the rank table record associated with the address of the selected layer k rank in said first file; and
- for any record of the filled rank table containing a layer k rank and associated with an address in the second file, reading the associated layer k bitmap segment at said address in the second file and superimposing the read layer k bitmap segment onto a segment of said layer k bitmap vector having a position determined by said layer k rank.

175. A method according to claim 171 or 172, wherein the step of determining the matching row identifier list further comprises, for any coding layer k such that $1 < k \leq n$, determining a layer k' filtering list for $k \leq k' \leq n$

WO 02/44943

PCT/IB01/02792

- 129 -

consisting of the layer k' input list obtained by providing the layer k result list as an input list in layer k of the coding scheme,

wherein, in the coding scheme, the coding data representing the position of each integer of an input list within a subset for a coding layer $k < n$ define a layer k bitmap segment in which each bit is associated with a respective integer of the subset to indicate whether said integer belongs to said input list, while a layer k integer rank associated with said layer k bitmap segment represents the position of said subset in the layer k pattern, and wherein the step of determining a layer k integer list for a leaf node for $k < n$ comprises:

10 /a/ initializing a layer k bitmap vector with logical zeroes;
 /b/ selecting the layer n ranks obtained from the coding data for each relevant thesaurus entry to which said leaf node corresponds, and setting $k' = n$;
 15 /c/ for each selected layer k' rank:
 /c1/ if the selected layer k' rank represents the position in the layer k' pattern of a subset which includes at least one integer of the layer k' filtering list, obtaining the layer k' bitmap segment with which the selected layer k' rank is associated;
 20 /c2/ for any integer of the layer k' filtering list whose position within said subset is represented in said layer k' bitmap segment, selecting a respective layer $k'-1$ rank determined from the selected layer k' rank and said position represented in said layer k' bitmap segment;
 25 /c3/ if $k' > k+1$, executing step /c/ with k' decremented by one unit; and
 /c4/ if $k'-1 = k$, obtaining any layer k bitmap segment with which a selected layer $k'-1$ rank is associated, and superimposing said layer k bitmap segment onto a segment of said layer k bitmap vector having a position determined by said selected layer $k'-1$ rank, the superimposition being performed according to a bitwise Boolean OR operation,
 30

WO 02/44943

PCT/IB01/02792

- 130 -

said layer k list for the leaf node corresponding to the resulting layer k bitmap vector.

176. A method according to claim 175, wherein, for $1 \leq k < n$, the layer k bitmap segments for at least one thesaurus entry are stored in at least one layer k file at addresses respectively corresponding to the layer k ranks associated therewith, and wherein, for $1 \leq k < n$, the step of determining a layer k integer list for a leaf node comprises:
- providing a rank table in a RAM memory, having records associated with the addresses in said layer k file;
 - 10 - filling the rank table by writing any selected layer k rank into the rank table record associated with the address corresponding to the selected layer k rank; and
 - for any record of the filled rank table containing a layer k rank and associated with an address in said layer k file, reading the associated layer k bitmap segment at said address and superimposing the read layer k bitmap segment onto a segment of said layer k bitmap vector having a position determined by said layer k rank.

177. A method according to any one of claims 151 to 176, wherein the step of processing the matching row identifier list comprises, for at least one attribute specified in the query, selecting a thesaurus associated with said attribute and detecting entries of the selected thesaurus with which identifier lists having a non-empty intersection with the matching row identifier list are associated.

178. A method according to claim 177, wherein said attribute specified in the query has $Q+1$ stored thesauruses associated with different prefix lengths, Q being an integer at least equal to 0, each of said $Q+1$ thesauruses having a thesaurus level parameter q such that $0 \leq q \leq Q$, whereby the prefix length is a decreasing function of the level parameter q and corresponds to a word length of said attribute for $q = 0$, wherein each of said $Q+1$ thesauruses is defined with reference to a respective partition into subsets each consisting of words beginning by a common prefix having the prefix length associated with said

WO 02/44943

PCT/IB01/02792

- 131 -

thesaurus, the entries of said thesaurus being sorted based on the prefix values.

179. A method according to claim 178, wherein, the selected thesaurus having a level parameter $QA \geq 0$, the detection of entries in the selected

5 thesaurus comprises the steps of:

/a/ providing respective level q target lists and respective level q thesaurus ranges covering consecutive entries of the level q thesaurus for $QA \leq q \leq Q$;

10 /b/ initializing the level Q target list with the matching row identifier list, initializing the level parameter q with the value Q , and selecting a first entry of the level Q thesaurus range;

/c/ determining an intersection list between the level q target list and the identifier list associated with the selected entry of the level q thesaurus range;

15 /d/ if the intersection list determined in the preceding step /c/ is empty, selecting another entry of the level q thesaurus range and repeating step /c/;

/e/ if q is greater than QA :

20 /e1/ setting the level $q-1$ target list as equal to the intersection list determined in the preceding step /c/;

/e2/ setting the level $q-1$ thesaurus range as consisting of the entries of the level $q-1$ thesaurus relating to level $q-1$ prefixes which begin with the level q prefix of the selected level q thesaurus entry, and selecting a first entry of the level $q-1$ thesaurus range;

25 /e3/ decrementing q by one unit and returning to step /c/;

/f/ if q is equal to QA :

/f1/ including the selected level QA thesaurus entry in the detected entries;

30 /f2/ if the level Q target list is equal to the intersection list determined in the preceding step /c/, terminating the detection of entries in the selected thesaurus;

WO 02/44943

PCT/IB01/02792

- 132 -

- /f3/ removing the integers of the intersection list determined in the preceding step /c/ from any target list including at least one integer which is not in said intersection list;
- /f4/ setting q as the smallest level parameter for which the target list includes at least one integer which is not in said intersection list;
- 5 /f5/ selecting another entry in the level q thesaurus range and returning to step /c/.

180. A method according to claim 179, wherein $Q \geq 1$ and each thesaurus having a level parameter $q \geq 1$ further contains, in each entry provided for a level q prefix value, data designating the entry of the level q-1 thesaurus which corresponds to the lowest or highest level q-1 prefix beginning with the level q prefix of said level q thesaurus entry, and wherein step /e2/ comprises selecting the level q-1 thesaurus entry designated in the selected level q thesaurus entry.

181. A method according to claim 179 or 180, wherein a coding scheme comprising n successive coding layers is used to provide coding data representing the identifier list associated with a level q thesaurus entry for $0 \leq q \leq Q$, n being a number at least equal to 1, each layer having a predetermined pattern for dividing a range covering integers of an input list of said layer into subsets, said identifier list being the input list of the first layer for said thesaurus entry, wherein for any layer other than the last layer, an integer list representing the position, in the pattern of said layer, of each subset containing at least one integer of the input list forms the input list for the next layer,

25 wherein the coding data comprise, for each layer and each subset containing at least one integer of the input list, data representing the position of each integer of the input list within said subset and, at least if said layer is the last layer, data representing the position of said subset in the pattern of said layer,

30 and wherein each level q target list forms a layer 1 and level q filtering list and is submitted as a layer 1 input list in the coding scheme for

WO 02/44943

PCT/IB01/02792

- 133 -

$QA \leq q \leq Q$ to provide respective layer k and level q filtering lists for $1 < k \leq n$ if $n > 1$, said layer k and level q filtering list provided from a level q target list being the layer k input list obtained from said level q target list in the coding scheme.

- 5 182. A method according to claim 181, wherein the pattern of each layer is such that the integer subsets are consecutive intervals consisting of the same number of integers.

183. A method according to claim 182, wherein said number of integers is a whole power of 2 for each layer.

- 10 184. A method according to any one of claims 181 to 183, wherein the step /c/ of determining the intersection list between a level q target list and an identifier list comprises, from $k = n$:

- /c1/ computing a layer k intersection list between the layer k input list
obtained from said identifier list in the coding scheme and the layer k
15 and level q filtering list corresponding to said level q target list;
/c2/ if the computed layer k intersection list is empty, determining said
intersection list between the level q target list and the identifier list as
being empty;
/c3/ if $k = 1$, determining said intersection list between the level q target list
20 and the identifier list as the computed layer 1 intersection list; and
/c4/ if $k > 1$, decrementing k by one unit and repeating from step /c1/.

185. A method according to claim 184, wherein, in the coding scheme,
the coding data representing the position of each integer of an input list within a
subset for a coding layer $k \leq n$ define a layer k bitmap segment in which each
25 bit is associated with a respective integer of the subset to indicate whether said
integer belongs to said input list, while the data representing the position of said
subset in the layer k pattern comprise a layer k integer rank associated with
said layer k bitmap segment, and wherein the step /c1/ of computing a layer k
intersection list between a layer k input list obtained from an identifier list in the

WO 02/44943

PCT/IB01/02792

- 134 -

coding scheme and a layer k and level q filtering list, represented by a first layer k bitmap vector, comprises:

- initializing a second layer k bitmap vector with logical zeroes;
 - obtaining layer k ranks from the coding data representing said identifier list; and
 - selecting any obtained layer k rank which represents the position in the layer k pattern of a subset including at least one integer of said layer k and level q filtering list, obtaining the layer k bitmap segment with which the selected layer k rank is associated, and determining a segment of the second layer k bitmap vector having a position determined by the selected layer k rank by combining the obtained layer k bitmap segment with a segment of the first layer k bitmap vector having a position determined by the selected layer k rank according to a bitwise Boolean AND operation,
- said layer k intersection list corresponding to the resulting second layer k bitmap vector.

186. A method according to claim 185, wherein, for $1 \leq k < n$, the layer k ranks and the layer k bitmap segments associated therewith for at least one thesaurus entry are stored at corresponding addresses in distinct first and second files, and wherein the step /c1/ of computing a layer k intersection list between a layer k input list obtained from an identifier list in the coding scheme and a layer k and level q filtering list, represented by a first layer k bitmap vector, comprises:

- providing a rank table in a RAM memory, having records associated with the addresses in said first and second files;
- filling the rank table by writing any selected layer k rank into the rank table record associated with the address of said selected layer k rank in said first file; and
- for any record of the filled rank table containing a layer k rank and associated with an address in the second file, reading the associated layer k bitmap segment at said address in the second file and combining the read layer k bitmap segment with a segment of the first layer k

WO 02/44943

PCT/IB01/02792

- 135 -

bitmap vector having a position determined by said layer k rank according to a bitwise Boolean AND operation to determine a segment of the second layer k bitmap vector having a position determined by said layer k rank.

- 5 187. A method according to any one of claims 181 to 183, wherein $n > 1$ and in the coding scheme, the coding data representing the position of each integer of an input list within a subset for a coding layer $k \leq n$ define a layer k bitmap segment in which each bit is associated with a respective integer of the subset to indicate whether said integer belongs to said input list, while a layer k
- 10 integer rank associated with said layer k bitmap segment represents the position of said subset in the layer k pattern, and wherein the step /c/ of determining the intersection list between a level q target list, corresponding to layer k and level q filtering lists represented by a respective first layer k bitmap vectors for $1 \leq k \leq n$, and an identifier list comprises:
- 15 /c1/ initializing a second bitmap vector with logical zeroes;
- /c2/ selecting layer n ranks obtained from the coding data representing said identifier list, and setting $k = n$;
- /c3/ for each selected layer k rank:
- 20 /c31/ if the selected layer k rank represents the position in the layer k pattern of a subset which includes at least one integer of said layer k and level q filtering list, obtaining the layer k bitmap segment with which the selected layer k rank is associated;
- 25 /c32/ for any integer of the layer k and level q filtering list whose position within said subset is represented in said layer k bitmap segment, selecting a respective layer $k-1$ rank determined from the selected layer k rank and said position represented in said layer k bitmap segment;
- /c33/ if $k > 2$, executing step /c3/ with k decremented by one unit;
- 30 and
- /c34/ if $k = 2$, obtaining any layer 1 bitmap segment with which a selected layer 1 rank is associated, and combining the

WO 02/44943

PCT/IB01/02792

- 136 -

obtained layer 1 bitmap segment with a segment of the first layer 1 bitmap vector having a position determined by said layer 1 rank according to a bitwise Boolean AND operation to determine a segment of the second bitmap vector having a position determined by said layer 1 rank, said intersection list corresponding to the resulting second bitmap vector.

188. A method according to claim 187, wherein the layer 1 bitmap segments for at least one thesaurus entry are stored in at least one layer 1 file at addresses respectively corresponding to the layer 1 ranks associated therewith, and the step /c/ of determining an intersection list comprises:

- providing a rank table in a RAM memory, having records associated with the addresses in said layer 1 file;
- filling the rank table by writing any layer 1 rank selected in step /c32/ into the rank table record associated with the address corresponding to the selected layer 1 rank; and
- for any record of the filled rank table containing a layer 1 rank and associated with an address in said layer 1 file, reading the associated layer 1 bitmap segment at said address and combining the read layer 1 bitmap segment with a segment of the first layer 1 bitmap vector having a position determined by said layer 1 rank according to a bitwise Boolean AND operation to determine a segment of the second layer 1 bitmap vector having a position determined by said layer 1 rank.

189. A method according to any one of claims 185 to 188, further comprising determining a pre-filtering flag for each entry of a level q thesaurus, said pre-filtering flag having a first value when said entry is associated with a reference table row identifier list represented by coding data which do not define any layer n rank representing the position in the layer n pattern of a subset which includes at least one integer of a layer n and level q filtering list, and wherein the step /c/ of determining the intersection list between a level q target list, corresponding to said layer n and level q filtering list, and an identifier list associated with an entry of the level q thesaurus comprises

WO 02/44943

PCT/IB01/02792

- 137 -

determining said intersection list as being empty if the pre-filtering flag determined for said entry has said first value.

190. A method according to claim 189, wherein, for any level q thesaurus entry associated with a row identifier list represented by coding data which
5 define a layer n rank representing the position in the layer n pattern of a subset which includes at least one integer of the layer n and level q filtering list, the layer n bitmap segment associated with said layer n rank is obtained and said first value is allocated to the pre-filtering flag determined for said entry if the
10 obtained layer n bitmap segment does not represent the position of any integer of said layer n and level q filtering list within said subset.

191. A method according to any one of claims 177 to 190, wherein the step of processing the matching row identifier list further comprises writing output data associated with any detected entry of a selected thesaurus into an output table.

15 192. A method according to claim 191, wherein the output table includes a respective row corresponding to each identifier of the matching row identifier list, and wherein output data associated with a detected entry of a selected thesaurus are written into any row of the output table corresponding to a
20 reference table row identifier belonging to both the matching row identifier list and the identifier list associated with said detected thesaurus entry.

193. A method according to claim 192, wherein each reference table row identifier has a respective row of the output table corresponding thereto, wherein the rows of the output table are initialized with a default value before
25 writing the output data, and wherein the rows of the output table which do not contain the default value are read after writing the output data.

194. A method according to claim 192, wherein the output table is associated with an index file having a respective record for each reference
table row identifier, containing either a default value or a pointer designating a
30 respective row of the output table corresponding to said reference table row identifier, wherein the records of the index file are initialized with a default value

WO 02/44943

PCT/IB01/02792

- 138 -

before writing the output data, and wherein the step of writing output data associated with a detected entry of a first selected thesaurus comprises, for each reference table row identifier belonging to both the matching row identifier list and the identifier list associated with said detected entry of the first selected

5 thesaurus:

- allocating an available row of the output table to correspond to said reference table row identifier,
 - writing output data into the allocated row; and
 - writing a pointer to the allocated row into the record of the index file
- 10 provided for said reference table row identifier.

195. A method according to any one of claims 192 to 194, wherein the output table has a plurality of columns each associated with a respective attribute for which a thesaurus is selected, and wherein output data associated with a detected entry of a thesaurus selected for an attribute associated with a

15 column of the output table are written into said column.

196. A method according to both claims 194 and 195, wherein the step of writing output data associated with a detected entry of at least one second selected thesaurus comprises, for each reference table row identifier belonging to both the matching row identifier list and the identifier list associated with said

20 detected entry of the second selected thesaurus:

- reading the pointer contained in the record of the index file provided for said reference table row identifier; and
- writing output data into the row of the output table designated by said pointer.

25 197. A method according to any one of claims 191 to 196, wherein the selected thesaurus being a word thesaurus defined with reference to a partition into subsets each consisting of one word, the output data associated with a detected entry comprise the word for which said detected entry is provided.

198. A method according to any one of claims 191 to 196, wherein the

30 selected thesaurus being a macroword thesaurus associated with a prefix

WO 02/44943

PCT/IB01/02792

- 139 -

length and defined with reference to a partition into subsets each consisting of words beginning by a common prefix having said prefix length, the output data associated with a detected entry comprise the prefix value for which said detected entry is provided.

5 199. A method according to any one of claims 191 to 196, wherein the output data associated with a detected entry comprise an address of said detected entry in the selected thesaurus.

200. A method according to any one of claims 191 to 196, wherein the output data associated with a detected entry of a selected thesaurus comprise
10 a numerical value derived from said thesaurus entry.

201. A method according to claim 200, wherein, for a detected entry of at least one selected thesaurus, said numerical value is calculated by applying a mathematical function to a thesaurus value stored in said entry.

202. A method according to claim 200 or 201, wherein, for a detected
15 entry of at least one selected thesaurus, said numerical value is calculated by applying a mathematical function to a plurality of values including a thesaurus value stored in said entry and at least one value already present in the output table.

203. A method according to any one of claims 200 to 202, wherein the
20 output table includes a respective row corresponding to each identifier of the matching row identifier list, and wherein a numerical value derived from a detected thesaurus entry is written into any row of the output table corresponding to a reference table row identifier belonging to both the matching row identifier list and the identifier list associated with said detected thesaurus
25 entry.

204. A method according to claim 203, wherein the numerical value, derived from a detected entry of a first selected thesaurus and written into any row of the output table corresponding to a reference table row identifier belonging to both the matching row identifier list and the identifier list

WO 02/44943

PCT/IB01/02792

- 140 -

associated with said entry of the first selected thesaurus, is obtained from a thesaurus value stored in said entry,

and wherein the numerical value, derived from a detected entry of at least one second selected thesaurus and written into a row of the output table
5 corresponding to a reference table row identifier belonging to both the matching row identifier list and the identifier list associated with said entry of the second selected thesaurus, is calculated by applying a mathematical function to a plurality of values including a thesaurus value stored in said entry and at least one value already present in said row of the output table.

10 205. A method according to claim 203 or 204, further comprising calculating an output value from a set of numerical values which have been respectively written into the rows of the output table.

206. A method according to any one of claims 151 to 205, wherein an integer range covering the identifiers designating the rows of the reference
15 table is partitioned into a plurality of predetermined portions, wherein at least some of the data representing identifier lists are distributed into a plurality of storage sections respectively associated with said portions, wherein a storage section associated with one of said portions contains data representing identifier sub-lists consisting of identifiers of said portion.

20 and wherein the step of determining a matching row identifier list is executed separately for the different portions of the reference table row identifier range, by means of the respective storage sections.

207. A method according to claim 206, wherein the step of processing the matching row identifier is at least partially executed separately for the different
25 portions of the reference table row identifier range, by means of the respective storage sections.

208. A method according to claim 206 or 207, wherein the thesauruses have a plurality of sections respectively associated with said portions, wherein a section, associated with one of said portions, of a thesaurus associated with
30 an attribute and defined with reference to a partition into subsets has a

WO 02/44943

PCT/IB01/02792

- 141 -

- respective entry for each subset of said partition which includes at least one word assigned to said attribute in a reference table row to which an identifier of said portion is allocated, said entry containing data representing an identifier sub-list including each identifier of said portion allocated to a reference table
- 5 row having a word of said subset assigned to said attribute, and wherein the step of analyzing the query criteria is at least partially executed separately for the different portions of the reference table row identifier range, by means of the respective thesaurus sections.
209. A method according to any one of claims 206 to 208, wherein the
- 10 separate step executions are carried out in parallel by respective processors for the different portions of the reference table row identifier range.
210. A method according to claim 209, wherein each thesaurus entry has a plurality of fields respectively associated with said portions, for containing data for retrieving respective identifier sub-lists from the storage sections,
- 15 wherein the step of analyzing the query criteria is executed centrally for all the portions of the reference table row identifier range, and wherein the relevant thesaurus entries used by a processor executing the step of determining a matching row identifier list by means of a storage section are designated by the data for retrieving identifier sub-lists from said storage section.
- 20 211. A method according to claim 210, wherein the step of analyzing the query criteria is executed by a query server connected to said processors through a communication network.
212. A method according to claim 211, wherein a list update server is connected, through the communication network, to a plurality of storage units
- 25 respectively coupled to said processors, the list update server controlling the storage units to maintain the storage sections.
213. A method according to any one of claims 151 to 212, wherein said reference table is a virtual table which is not stored.

WO 02/44943

PCT/IB01/02792

- 142 -

214. A method of processing a query in a database system, wherein a plurality of row identifiers are defined to designate respective rows of a reference table having columns respectively associated with data attributes, said rows containing groups of related words assigned to said attributes in a collection of data, wherein a plurality of thesauruses each associated with a respective attribute and data representing reference table row identifier lists respectively associated with entries of said thesauruses are stored, wherein each thesaurus associated with one attribute is defined with reference to a partition into subsets of a set of words which can be assigned to said one attribute and has a respective entry for each subset including at least one word assigned to said one attribute in the collection of data, the reference table row identifier list associated with said thesaurus entry including any identifier allocated to a row of the reference table having a word of said subset assigned to said one attribute, the method comprising the steps of:
- determining a matching reference table row identifier list based on a combination of thesaurus entries relevant to the query and on the stored data representing the reference table row identifier lists associated with said relevant thesaurus entries; and
 - processing said matching row identifier list to output a response,
- wherein the step of processing the matching row identifier list comprises, for at least one attribute specified in the query, selecting a thesaurus associated with said attribute and detecting entries of the selected thesaurus with which identifier lists having a non-empty intersection with the matching row identifier list are associated.
215. A method according to claim 214, wherein said attribute specified in the query has $Q+1$ stored thesauruses associated with different prefix lengths, Q being an integer at least equal to 0, each of said $Q+1$ thesauruses having a thesaurus level parameter q such that $0 \leq q \leq Q$, whereby the prefix length is a decreasing function of the level parameter q and corresponds to a word length of said attribute for $q = 0$, wherein each of said $Q+1$ thesauruses is defined with reference to a respective partition into subsets each consisting of words beginning by a common prefix having the prefix length associated with said

WO 02/44943

PCT/IB01/02792

- 143 -

thesaurus, the entries of said thesaurus being sorted based on the prefix values.

216. A method according to claim 215, wherein, the selected thesaurus having a level parameter $QA \geq 0$, the detection of entries in the selected thesaurus comprises the steps of:
- 5 /a/ providing respective level q target lists and respective level q thesaurus ranges covering consecutive entries of the level q thesaurus for $QA \leq q \leq Q$;
 - /b/ initializing the level Q target list with the matching row identifier list,
 - 10 initializing the level parameter q with the value Q, and selecting a first entry of the level Q thesaurus range;
 - /c/ determining an intersection list between the level q target list and the identifier list associated with the selected entry of the level q thesaurus range;
 - 15 /d/ if the intersection list determined in the preceding step /c/ is empty, selecting another entry of the level q thesaurus range and repeating step /c/;
 - /e/ if q is greater than QA:
 - /e1/ setting the level q-1 target list as equal to the intersection list
 - 20 determined in the preceding step /c/;
 - /e2/ setting the level q-1 thesaurus range as consisting of the entries of the level q-1 thesaurus relating to level q-1 prefixes which begin with the level q prefix of the selected level q thesaurus entry, and selecting a first entry of the level q-1 thesaurus
 - 25 range;
 - /e3/ decrementing q by one unit and returning to step /c/;
 - /f/ if q is equal to QA:
 - /f1/ including the selected level QA thesaurus entry in the detected
 - entries;
 - 30 /f2/ if the level Q target list is equal to the intersection list determined in the preceding step /c/, terminating the detection of entries in the selected thesaurus;

WO 02/44943

PCT/IB01/02792

- 144 -

- /f3/ removing the integers of the intersection list determined in the preceding step /c/ from any target list including at least one integer which is not in said intersection list;
 /f4/ setting q as the smallest level parameter for which the target list includes at least one integer which is not in said intersection list;
 5 /f5/ selecting another entry in the level q thesaurus range and returning to step /c/.

217. A method according to claim 216, wherein $Q \geq 1$ and each thesaurus having a level parameter $q \geq 1$ further contains, in each entry provided for a level q prefix value, data designating the entry of the level q-1 thesaurus which corresponds to the lowest or highest level q-1 prefix beginning with the level q prefix of said level q thesaurus entry, and wherein step /e2/ comprises selecting the level q-1 thesaurus entry designated in the selected level q thesaurus entry.

218. A method according to claim 216 or 217, wherein a coding scheme comprising n successive coding layers is used to provide coding data representing the identifier list associated with a level q thesaurus entry for $0 \leq q \leq Q$, n being a number at least equal to 1, each layer having a predetermined pattern for dividing a range covering integers of an input list of said layer into subsets, said identifier list being the input list of the first layer for said thesaurus entry, wherein for any layer other than the last layer, an integer list representing the position, in the pattern of said layer, of each subset containing at least one integer of the input list forms the input list for the next layer,

25 wherein the coding data comprise, for each layer and each subset containing at least one integer of the input list, data representing the position of each integer of the input list within said subset and, at least if said layer is the last layer, data representing the position of said subset in the pattern of said layer,

30 and wherein each level q target list forms a layer 1 and level q filtering list and is submitted as a layer 1 input list in the coding scheme for

WO 02/44943

PCT/IB01/02792

- 145 -

$QA \leq q \leq Q$ to provide respective layer k and level q filtering lists for $1 < k \leq n$ if $n > 1$, said layer k and level q filtering list provided from a level q target list being the layer k input list obtained from said level q target list in the coding scheme.

- 5 219. A method according to claim 218, wherein the pattern of each layer is such that the integer subsets are consecutive intervals consisting of the same number of integers.

220. A method according to claim 219, wherein said number of integers is a whole power of 2 for each layer.

- 10 221. A method according to any one of claims 218 to 220, wherein the step /c/ of determining the intersection list between a level q target list and an identifier list comprises, from $k = n$:

- /c1/ computing a layer k intersection list between the layer k input list
obtained from said identifier list in the coding scheme and the layer k
15 and level q filtering list corresponding to said level q target list;
/c2/ if the computed layer k intersection list is empty, determining said
intersection list between the level q target list and the identifier list as
being empty;
/c3/ if $k = 1$, determining said intersection list between the level q target list
20 and the identifier list as the computed layer 1 intersection list; and
/c4/ if $k > 1$, decrementing k by one unit and repeating from step /c1/.

222. A method according to claim 221, wherein, in the coding scheme,
the coding data representing the position of each integer of an input list within a
subset for a coding layer $k \leq n$ define a layer k bitmap segment in which each
25 bit is associated with a respective integer of the subset to indicate whether said
integer belongs to said input list, while the data representing the position of said
subset in the layer k pattern comprise a layer k integer rank associated with
said layer k bitmap segment, and wherein the step /c1/ of computing a layer k
intersection list between a layer k input list obtained from an identifier list in the

WO 02/44943

PCT/IB01/02792

- 146 -

coding scheme and a layer k and level q filtering list, represented by a first layer k bitmap vector, comprises:

- initializing a second layer k bitmap vector with logical zeroes;
 - obtaining layer k ranks from the coding data representing said identifier list; and
 - selecting any obtained layer k rank which represents the position in the layer k pattern of a subset including at least one integer of said layer k and level q filtering list, obtaining the layer k bitmap segment with which the selected layer k rank is associated, and determining a segment of the second layer k bitmap vector having a position determined by the selected layer k rank by combining the obtained layer k bitmap segment with a segment of the first layer k bitmap vector having a position determined by the selected layer k rank according to a bitwise Boolean AND operation,
- said layer k intersection list corresponding to the resulting second layer k bitmap vector.

223. A method according to claim 222, wherein, for $1 \leq k < n$, the layer k ranks and the layer k bitmap segments associated therewith for at least one thesaurus entry are stored at corresponding addresses in distinct first and second files, and wherein the step /c1/ of computing a layer k intersection list between a layer k input list obtained from an identifier list in the coding scheme and a layer k and level q filtering list, represented by a first layer k bitmap vector, comprises:

- providing a rank table in a RAM memory, having records associated with the addresses in said first and second files;
- filling the rank table by writing any selected layer k rank into the rank table record associated with the address of said selected layer k rank in said first file; and
- for any record of the filled rank table containing a layer k rank and associated with an address in the second file, reading the associated layer k bitmap segment at said address in the second file and combining the read layer k bitmap segment with a segment of the first layer k

WO 02/44943

PCT/IB01/02792

- 147 -

bitmap vector having a position determined by said layer k rank according to a bitwise Boolean AND operation to determine a segment of the second layer k bitmap vector having a position determined by said layer k rank.

- 5 224. A method according to any one of claims 218 to 220, wherein $n > 1$ and in the coding scheme, the coding data representing the position of each integer of an input list within a subset for a coding layer $k \leq n$ define a layer k bitmap segment in which each bit is associated with a respective integer of the subset to indicate whether said integer belongs to said input list, while a layer k
- 10 integer rank associated with said layer k bitmap segment represents the position of said subset in the layer k pattern, and wherein the step /c/ of determining the intersection list between a level q target list, corresponding to layer k and level q filtering lists represented by a respective first layer k bitmap vectors for $1 \leq k \leq n$, and an identifier list comprises:
- 15 /c1/ initializing a second bitmap vector with logical zeroes;
- /c2/ selecting layer n ranks obtained from the coding data representing said identifier list, and setting $k = n$;
- /c3/ for each selected layer k rank:
- 20 /c31/ if the selected layer k rank represents the position in the layer k pattern of a subset which includes at least one integer of said layer k and level q filtering list, obtaining the layer k bitmap segment with which the selected layer k rank is associated;
- 25 /c32/ for any integer of the layer k and level q filtering list whose position within said subset is represented in said layer k bitmap segment, selecting a respective layer $k-1$ rank determined from the selected layer k rank and said position represented in said layer k bitmap segment;
- /c33/ if $k > 2$, executing step /c3/ with k decremented by one unit;
- 30 and
- /c34/ if $k = 2$, obtaining any layer 1 bitmap segment with which a selected layer 1 rank is associated, and combining the

WO 02/44943

PCT/IB01/02792

- 148 -

obtained layer 1 bitmap segment with a segment of the first layer 1 bitmap vector having a position determined by said layer 1 rank according to a bitwise Boolean AND operation to determine a segment of the second bitmap vector having a position determined by said layer 1 rank,
 5 said intersection list corresponding to the resulting second bitmap vector.

225. A method according to claim 224, wherein the layer 1 bitmap segments for at least one thesaurus entry are stored in at least one layer 1 file at addresses respectively corresponding to the layer 1 ranks associated
 10 therewith, and the step /c/ of determining an intersection list comprises:

- providing a rank table in a RAM memory, having records associated with the addresses in said layer 1 file;
- filling the rank table by writing any layer 1 rank selected in step /c32/ into the rank table record associated with the address corresponding to the
 15 selected layer 1 rank; and
- for any record of the filled rank table containing a layer 1 rank and associated with an address in said layer 1 file, reading the associated layer 1 bitmap segment at said address and combining the read layer 1 bitmap segment with a segment of the first layer 1 bitmap vector having
 20 a position determined by said layer 1 rank according to a bitwise Boolean AND operation to determine a segment of the second layer 1 bitmap vector having a position determined by said layer 1 rank.

226. A method according to any one of claims 222 to 225, further comprising determining a pre-filtering flag for each entry of a level q thesaurus,
 25 said pre-filtering flag having a first value when said entry is associated with a reference table row identifier list represented by coding data which do not define any layer n rank representing the position in the layer n pattern of a subset which includes at least one integer of a layer n and level q filtering list, and wherein the step /c/ of determining the intersection list between a level q
 30 target list, corresponding to said layer n and level q filtering list, and an identifier list associated with an entry of the level q thesaurus comprises

WO 02/44943

PCT/IB01/02792

- 149 -

determining said intersection list as being empty if the pre-filtering flag determined for said entry has said first value.

227. A method according to claim 226, wherein, for any level q thesaurus entry associated with a row identifier list represented by coding data which
5 define a layer n rank representing the position in the layer n pattern of a subset which includes at least one integer of the layer n and level q filtering list, the layer n bitmap segment associated with said layer n rank is obtained and said first value is allocated to the pre-filtering flag determined for said entry if the
10 obtained layer n bitmap segment does not represent the position of any integer of said layer n and level q filtering list within said subset.

228. A method according to any one of claims 214 to 227, wherein the step of processing the matching row identifier list further comprises writing output data associated with any detected entry of a selected thesaurus into an output table.

15 229. A method according to claim 228, wherein the output table includes a respective row corresponding to each identifier of the matching row identifier list, and wherein output data associated with a detected entry of a selected thesaurus are written into any row of the output table corresponding to a
20 reference table row identifier belonging to both the matching row identifier list and the identifier list associated with said detected thesaurus entry.

230. A method according to claim 229, wherein each reference table row identifier has a respective row of the output table corresponding thereto, wherein the rows of the output table are initialized with a default value before
25 writing the output data, and wherein the rows of the output table which do not contain the default value are read after writing the output data.

231. A method according to claim 229, wherein the output table is associated with an index file having a respective record for each reference
table row identifier, containing either a default value or a pointer designating a
respective row of the output table corresponding to said reference table row
30 identifier, wherein the records of the index file are initialized with a default value

WO 02/44943

PCT/IB01/02792

- 150 -

before writing the output data, and wherein the step of writing output data associated with a detected entry of a first selected thesaurus comprises, for each reference table row identifier belonging to both the matching row identifier list and the identifier list associated with said detected entry of the first selected

5 thesaurus:

- allocating an available row of the output table to correspond to said reference table row identifier,
 - writing output data into the allocated row; and
 - writing a pointer to the allocated row into the record of the index file
- 10 provided for said reference table row identifier.

232. A method according to any one of claims 229 to 231, wherein the output table has a plurality of columns each associated with a respective attribute for which a thesaurus is selected, and wherein output data associated with a detected entry of a thesaurus selected for an attribute associated with a column of the output table are written into said column.

15

233. A method according to both claims 231 and 232, wherein the step of writing output data associated with a detected entry of at least one second selected thesaurus comprises, for each reference table row identifier belonging to both the matching row identifier list and the identifier list associated with said detected entry of the second selected thesaurus:

20

- reading the pointer contained in the record of the index file provided for said reference table row identifier; and
- writing output data into the row of the output table designated by said pointer.

25 234. A method according to any one of claims 228 to 233, wherein the selected thesaurus being a word thesaurus defined with reference to a partition into subsets each consisting of one word, the output data associated with a detected entry comprise the word for which said detected entry is provided.

235. A method according to any one of claims 228 to 233, wherein the selected thesaurus being a macroword thesaurus associated with a prefix

30

WO 02/44943

PCT/IB01/02792

- 151 -

length and defined with reference to a partition into subsets each consisting of words beginning by a common prefix having said prefix length, the output data associated with a detected entry comprise the prefix value for which said detected entry is provided.

5 236. A method according to any one of claims 228 to 233, wherein the output data associated with a detected entry comprise an address of said detected entry in the selected thesaurus.

237. A method according to any one of claims 228 to 233, wherein the output data associated with a detected entry of a selected thesaurus comprise
10 a numerical value derived from said thesaurus entry.

238. A method according to claim 237, wherein, for a detected entry of at least one selected thesaurus, said numerical value is calculated by applying a mathematical function to a thesaurus value stored in said entry.

239. A method according to claim 237 or 238, wherein, for a detected
15 entry of at least one selected thesaurus, said numerical value is calculated by applying a mathematical function to a plurality of values including a thesaurus value stored in said entry and at least one value already present in the output table.

240. A method according to any one of claims 237 to 239, wherein the
20 output table includes a respective row corresponding to each identifier of the matching row identifier list, and wherein a numerical value derived from a detected thesaurus entry is written into any row of the output table corresponding to a reference table row identifier belonging to both the matching row identifier list and the identifier list associated with said detected thesaurus
25 entry.

241. A method according to claim 240, wherein the numerical value, derived from a detected entry of a first selected thesaurus and written into any row of the output table corresponding to a reference table row identifier belonging to both the matching row identifier list and the identifier list

WO 02/44943

PCT/IB01/02792

- 152 -

associated with said entry of the first selected thesaurus, is obtained from a thesaurus value stored in said entry,

and wherein the numerical value, derived from a detected entry of at least one second selected thesaurus and written into a row of the output table
5 corresponding to a reference table row identifier belonging to both the matching row identifier list and the identifier list associated with said entry of the second selected thesaurus, is calculated by applying a mathematical function to a plurality of values including a thesaurus value stored in said entry and at least one value already present in said row of the output table.

10 242. A method according to claim 240 or 241, further comprising calculating an output value from a set of numerical values which have been respectively written into the rows of the output table.

243. A method according to any one of claims 214 to 242, wherein an integer range covering the identifiers designating the rows of the reference
15 table is partitioned into a plurality of predetermined portions, wherein at least some of the data representing identifier lists are distributed into a plurality of storage sections respectively associated with said portions, wherein a storage section associated with one of said portions contains data representing identifier sub-lists consisting of identifiers of said portion.

20 and wherein the step of determining a matching row identifier list is executed separately for the different portions of the reference table row identifier range, by means of the respective storage sections.

244. A method according to claim 243, wherein the step of processing the matching row identifier is at least partially executed separately for the different
25 portions of the reference table row identifier range, by means of the respective storage sections.

245. A method according to claim 243 or 244, wherein the separate step executions are carried out in parallel by respective processors for the different portions of the reference table row identifier range.

WO 02/44943

PCT/IB01/02792

- 153 -

246. A method according to claim 245, wherein the combination of thesaurus entries relevant to the query is determined, on the basis of criteria specified in the query, by a query server connected to said processors through a communication network.

5 247. A method according to claim 246, wherein a list update server is connected, through the communication network, to a plurality of storage units respectively coupled to said processors, the list update server controlling the storage units to maintain the storage sections.

248. A method according to any one of claims 214 to 247, wherein said
10 reference table is a virtual table which is not stored.

249. A method of encoding integer lists in a computer system, comprising the steps of:

- dividing a range covering integers of an input list into subsets according to a predetermined pattern; and
- 15 - producing coding data including, for each subset containing at least one integer of the input list, data representing the position of said subset in the pattern, and data representing the position of each integer of the input list within said subset.

250. A method according to claim 249, wherein the data representing the
20 position of each integer of the input list within a subset consist of a bitmap segment in which each bit is associated with a respective integer of the subset to indicate whether said integer belongs to the input list.

251. The method of claim 250, wherein the position of each subset in the pattern is represented by an integer rank which is included in the coding data,
25 in association with the corresponding bitmap segment, if said subset contains at least one integer of the input list.

252. A method according to claim 251, wherein a coding data container comprising records having respective addresses is provided for storing together coding data produced from a plurality of integer lists, wherein each record of

WO 02/44943

PCT/IB01/02792

- 154 -

the coding data container includes a first field for storing an integer rank related to the pattern, a second field for storing an address value and a third field for storing a bitmap segment, and wherein the encoding of a non-empty input list comprises the steps of:

- 5 a/ selecting an available record of the coding data container;
- b/ selecting a subset containing at least one integer of the input list to which no record has been allocated;
- c/ allocating the selected record to the selected subset;
- d/ storing the rank and the bitmap segment of the coding data produced for
10 the selected subset in the first and third fields of the selected record, respectively;
- e/ if every subset containing at least one integer of the input list has a record allocated thereto, storing an end value in the second field of the selected record; and
- 15 f/ if at least one subset containing at least one integer of the input list has no record allocated thereto, storing the address of an available record of the coding data container in the second field of the selected record, selecting said available record and repeating from step b/.

253. A method according to claim 252, wherein the coding data container
20 has a first file comprising the first and second fields of the records and a second file comprising the third fields of the records, the first and second files being accessible separately.

254. A method according to claim 252 or 253, further comprising the step
of grouping the records stored in the data container, so that the records
25 allocated to the subsets for any encoded integer list have contiguous addresses.

255. A method according to any one of claims 249 to 252, wherein the coding data produced from one integer list are stored in at least one file allocated to said one integer list.

WO 02/44943

PCT/IB01/02792

- 155 -

256. A method according to claim 255, wherein the coding data are stored in first and second files having a common addressing, whereby for each subset containing at least one integer of the input list, the data representing the position of said subset in the pattern are stored in the first file and the data
5 representing the position of each integer of the input list within said subset are stored at a corresponding address in the second file.

257. A method according to any one of claims 249 to 256, wherein the subsets are consecutive intervals consisting of the same number of integers.

258. A method according to claim 257, wherein said number of integers is
10 a whole power of 2.

259. A method of encoding integer lists in a computer system, comprising n successive coding layers, n being a number at least equal to 1, wherein each coding layer comprises the steps of:

- 15 - dividing a range covering integers of an input list of said layer into subsets according to a predetermined pattern;
- producing coding data including, for each subset containing at least one integer of the input list, data representing the position of each integer of the input list within said subset and, at least if said layer is the last coding layer, data representing the position of said subset in the pattern;
- 20 - if said layer is not the last coding layer, forming a further integer list representing the position, in the pattern of said layer, of each subset containing at least one integer of the input list, and providing said further integer list as an input list of the next layer.

260. A method according to claim 259, wherein, in the pattern of each
25 layer, the subsets are consecutive intervals consisting of the same number of integers.

261. A method according to claim 260, wherein said number of integers is a whole power of 2 for each layer.

WO 02/44943

PCT/IB01/02792

- 156 -

262. A method according to any one of claims 259 to 261, wherein the coding data produced for each layer are stored in first and second files having a common addressing, whereby for each subset containing at least one integer of the input list of said layer, the data representing the position of said subset in the pattern are stored in the first file and the data representing the position of each integer of the input list within said subset are stored at a corresponding address in the second file.
263. A method according to any one of claims 259 to 262, wherein the coding data produced from one integer list input in the first layer are stored in at least one file allocated to said one integer list.
264. A method according to any one of claims 259 to 262, wherein the coding data produced from one integer list input in the first layer are stored as at least one record chain in a data container allocated to a plurality of integer lists.
265. A method according to claim 264, further comprising the step of grouping the records of the data container so that the records of each chain have contiguous addresses.
266. A method according to any one of claims 259 to 265, wherein $n \geq 2$ and layer k data containers each having a plurality of records are provided in a computer memory for $1 \leq k \leq n$, each record of a layer k data container being associated with a layer k integer rank representing the position of a subset in the layer k pattern, and wherein each record of a layer k data container associated with a layer k rank representing the position of a subset in the layer k pattern has a first field for containing data for retrieving the position within said subset of any integer of a layer k input list relating to a layer 1 input list, whereby a combination of said layer k rank with any position retrievable from the data contained in said first field determines a layer $k-1$ rank with which a respective record of the layer $k-1$ data container is associated if $k > 1$, and an integer of said layer 1 input list if $k = 1$.

WO 02/44943

PCT/IB01/02792

- 157 -

267. A method according to claim 266, wherein each record of the layer n data container associated with a layer n rank further has a second field for containing said layer n rank.
268. A method according to claim 266 or 267, wherein, for $1 \leq k \leq n$, said data contained in the first field of a record of the layer k data container for retrieving the position of any integer of a layer k input list within a subset comprise a bitmap segment in which each bit is associated with a respective integer of said subset to indicate whether said integer belongs to said layer k input list.
- 10 269. A method according to claim 268, wherein, for $1 \leq k \leq n$, each record of the layer k data container associated with a layer k rank further has a second field for containing said layer k rank.
270. A method according to claim 269, wherein each data container comprises at least two files where the first and second fields of the records of said data container are respectively stored, said files being accessible separately.
- 15 271. A method according to claim 266 or 267, wherein, for $1 \leq k < n$, each record of the layer k data container further has a second field for containing a number representing the position of an integer of a layer k+1 input list within a subset of the layer k+1 pattern,
- 20 and wherein, for $1 < k \leq n$, said data contained in the first field of a record of the layer k data container associated with a layer k rank for retrieving the position of any integer of a layer k input list within a subset of the layer k pattern comprise a pointer to at least one record of the layer k-1 data container
- 25 in which the second field contains a number representing the position of an integer of said layer k input list within said subset of the layer k pattern, whereby said record of the layer k-1 data container is associated with the layer k-1 rank determined by the combination of said layer k rank with the position represented by said number.

WO 02/44943

PCT/IB01/02792

- 158 -

272. A method according to claim 271, wherein said data contained in the first field of a record of the layer 1 data container for retrieving the position of any integer of a layer 1 input list within a subset comprise a bitmap segment in which each bit is associated with a respective integer of said subset to indicate whether said integer belongs to said layer 1 input list.

273. A method according to claim 271 or 272, wherein each layer k data container for $1 \leq k < n$ comprises at least two files where the first and second fields of the records of said data container are respectively stored, said files being accessible separately.

274. A method according to any one of claims 266 to 273, wherein, for $1 \leq k \leq n$, each record of the layer k data container further has a next address field, whereby record chains are defined in the layer k data container by means of the next address fields, and wherein at least some of the layer 1 input lists are respectively associated with record chains in the layer n data container, whereby the coding data for layer n relating to one of said layer 1 input lists are stored in or retrievable from the record chain associated therewith in the layer n data container.

275. A method according to claim 274, wherein, for $1 \leq k < n$, said layer 1 input lists are respectively associated with record chains in the layer k data container, whereby the coding data relating to one of said layer 1 input lists for layer k are stored in or retrievable from the record chain associated therewith in the layer k data container.

276. A method according to claim 274, wherein, for $1 < k \leq n$, each record of the layer k data container further has a head address field for pointing to an address of a first record of a respective chain in the layer k-1 data container.

277. A method according to any one of claims 274 to 276, wherein each layer k data container for $1 \leq k \leq n$ comprises at least two files where the first fields and the next address fields of the records of said data container are respectively stored, said files being accessible separately.

WO 02/44943

PCT/IB01/02792

- 159 -

278. A method according to any one of claims 274 to 277, further comprising the step of grouping the records of the data container for each coding layer, so that the records of each chain have contiguous addresses.

279. A computerized method of combining a plurality of first integer lists into a second integer list, wherein at least one of the first integer lists is represented by stored coding data provided by a coding scheme comprising n successive coding layers, n being a number at least equal to 1, each layer having a predetermined pattern for dividing a range covering integers of an input list of said layer into subsets, said first integer list being the input list of the first layer, wherein for any layer other than the last layer, an integer list representing the position, in the pattern of said layer, of each subset containing at least one integer of the input list forms the input list for the next layer, wherein the stored coding data representing a first integer list comprise, for each layer and each subset containing at least one integer of the input list, data representing the position of each integer of the input list within said subset and, at least if said layer is the last layer, data representing the position of said subset in the pattern of said layer, the method comprising the steps of:

- defining a combination of intermediary lists each corresponding to at least one of the first integer lists;
- for k decreasing from n to 1, computing a layer k result list by combining a plurality of layer k intermediary lists in accordance with said combination; and
- producing the second integer list as the layer 1 result list,

and wherein, for any intermediary list corresponding to at least one first integer list represented by stored coding data, the layer n intermediary list is determined from said stored coding data as consisting of the integers of any layer n input list associated with said at least one first integer list in the coding scheme and, if $n > 1$, each layer k intermediary list for $k < n$ is determined from said stored coding data and the layer k+1 result list as consisting of any integer of a layer k input list associated with said at least one first integer list in the coding scheme which belongs to a layer k subset whose position is represented in the layer k+1 result list.

WO 02/44943

PCT/IB01/02792

- 160 -

280. A method according to claim 279, wherein, in the pattern of each layer, the subsets are consecutive intervals consisting of the same number of integers.

281. A method according to claim 280, wherein said number of integers is
5 a whole power of 2 for each layer.

282. A method according to any one of claims 279 to 281, wherein, in the coding scheme, the coding data representing the position of each integer of an input list within a subset for the coding layer n define a layer n bitmap segment in which each bit is associated with a respective integer of the subset to
10 indicate whether said integer belongs to said input list, while the data representing the position of said subset in the layer n pattern comprise a layer n integer rank associated with said layer n bitmap segment, and wherein the layer n intermediary list for an intermediary list corresponding to at least one first integer list represented by stored coding data is determined in a procedure
15 comprising:

- initializing a layer n bitmap vector with logical zeroes;
- obtaining the layer n ranks and associated bitmap segments from said stored coding data; and
- for each of said layer n ranks, superimposing the layer n bitmap segment
20 associated therewith onto a segment of said layer n bitmap vector having a position determined by said layer n rank, the superimposition being performed according to a bitwise Boolean OR operation,
said layer n intermediary list corresponding to the resulting layer n bitmap vector.

283. A method according to claim 282, wherein $n > 1$ and in the coding scheme, the coding data representing the position of each integer of an input list within a subset for a coding layer $k < n$ define a layer k bitmap segment in which each bit is associated with a respective integer of the subset to indicate whether said integer belongs to said input list, while the coding data further
30 comprise a layer k integer rank associated with said layer k bitmap segment to represent the position of said subset in the layer k pattern, and wherein, for

WO 02/44943

PCT/IB01/02792

- 161 -

$k < n$, the layer k intermediary list for an intermediary list corresponding to at least one first integer list represented by stored coding data is determined in a procedure comprising:

- initializing a layer k bitmap vector with logical zeroes;
- 5 - obtaining the layer k ranks from said stored coding data; and
- selecting any obtained layer k rank belonging to the layer $k+1$ result list and superimposing the associated layer k bitmap segment onto a segment of said layer k bitmap vector having a position determined by the selected layer k rank, the superimposition being performed according to a bitwise Boolean OR operation,
- 10 said layer k intermediary list corresponding to the resulting layer k bitmap vector.

284. A method according to claim 283, wherein, for $1 \leq k < n$, the layer k ranks and the layer k bitmap segments associated therewith are stored at corresponding addresses in distinct first and second files, and said procedure for determining the layer k intermediary list for an intermediary list corresponding to at least one first integer list represented by stored coding data comprises:

- providing a rank table in a RAM memory, having records associated with the addresses in said first and second files;
- 20 - filling the rank table by writing any selected layer k rank into the rank table record associated with the address of the selected layer k rank in said first file; and
- for any record of the filled rank table containing a layer k rank and associated with an address in the second file, reading the associated layer k bitmap segment at said address in the second file and superimposing the read layer k bitmap segment onto a segment of said layer k bitmap vector having a position determined by said layer k rank.
- 25

285. A method according to claim 282, wherein $n > 1$ and for any coding layer k such that $1 < k \leq n$, a layer k' filtering list is determined for $k \leq k' \leq n$,

30

WO 02/44943

PCT/IB01/02792

- 162 -

said layer k' filtering list being the layer k' input list obtained by providing the layer k result list as an input list in layer k of the coding scheme,

wherein, in the coding scheme, the coding data representing the position of each integer of an input list within a subset for a coding layer $k < n$ define a layer k bitmap segment in which each bit is associated with a respective integer of the subset to indicate whether said integer belongs to said input list, while a layer k integer rank associated with said layer k bitmap segment represents the position of said subset in the layer k pattern, and wherein, for $k < n$, the layer k intermediary list for an intermediary list corresponding to at least one first integer list represented by stored coding data is determined in a procedure comprising:

/a/ initializing a layer k bitmap vector with logical zeroes;
/b/ selecting the layer n ranks obtained from said stored coding data, and setting $k' = n$;
 15 */c/* for each selected layer k' rank:
 /c1/ if the selected layer k' rank represents the position in the layer k' pattern of a subset which includes at least one integer of the layer k' filtering list, obtaining the layer k' bitmap segment with which the selected layer k' rank is associated;
 20 */c2/* for any integer of the layer k' filtering list whose position within said subset is represented in said layer k' bitmap segment, selecting a respective layer $k'-1$ rank determined from the selected layer k' rank and said position represented in said layer k' bitmap segment;
 25 */c3/* if $k' > k+1$, executing step */c/* with k' decremented by one unit; and
 /c4/ if $k'-1 = k$, obtaining any layer k bitmap segment with which a selected layer $k'-1$ rank is associated, and superimposing said layer k bitmap segment onto a segment of said layer k bitmap vector having a position determined by said selected layer $k'-1$ rank, the superimposition being performed according to a bitwise Boolean OR operation,
 30

WO 02/44943

PCT/IB01/02792

- 163 -

said layer k intermediary list corresponding to the resulting layer k bitmap vector.

286. A method according to claim 285, wherein, for $1 \leq k < n$, the layer k bitmap segments are stored in at least one layer k file at addresses respectively corresponding to the layer k ranks associated therewith, and said procedure for determining the layer k intermediary list for an intermediary list corresponding to at least one first integer list represented by stored coding data comprises:

- providing a rank table in a RAM memory, having records associated with the addresses in said layer k file;
- filling the rank table by writing any selected layer k rank into the rank table record associated with the address corresponding to the selected layer k rank; and
- for any record of the filled rank table containing a layer k rank and associated with an address in said layer k file, reading the associated layer k bitmap segment at said address and superimposing the read layer k bitmap segment onto a segment of said layer k bitmap vector having a position determined by said layer k rank.

287. A method according to any one of claims 279 to 284, wherein a coding data container comprising records having respective addresses is provided for each coding layer $k \leq n$, for storing together layer k coding data of a plurality of said first integer lists, and wherein each record of the coding data container for each layer includes a first field for storing a rank related to the pattern of said layer, a second field for storing an address value and a third field for storing a bitmap segment, whereby said address value either points to another record of the data container where further layer k coding data relating to the same first integer list are stored or designates an end of coding data.

288. A method according to claim 287, wherein the records stored in the data container for each coding layer k are so grouped that the records where the layer k coding data of any first integer list are stored have contiguous addresses.

WO 02/44943

PCT/IB01/02792

- 164 -

289. A method according to any one of claims 279 to 286, wherein the coding data are stored in at least one file allocated to one first integer list.

290. A method according to claim 289, wherein the coding data of each layer are stored in first and second files having a common addressing, whereby
5 for each subset containing at least one integer of the input list of said layer, the data representing the position of said subset in the pattern are stored in the first file and the data representing the position of each integer of the input list within said subset are stored at a corresponding address in the second file.

291. A method according to any one of claims 279 to 290, wherein the
10 intermediary lists include at least one preset list, said preset list consisting of one of the first integer lists for which the layer k input lists, according to the coding scheme, are determined in advance for $1 \leq k \leq n$, said layer k input lists being the respective layer k intermediary lists corresponding to said preset list.

292. A database system for managing information from a collection of
15 data, comprising means arranged and programmed to implement a method as claimed in any one of claims 1 to 248.

293. A computer system, comprising means arranged and programmed to implement a method as claimed in any one of claims 249 to 291.

294. A computer program product, loadable into the internal memory of a
20 digital computer, comprising software code portions for carrying out a method as claimed in any one of claims 1 to 291 when said product is run on the computer.

WO 02/44943

PCT/IB01/02792

1/57

CLIENTS

Row-ID	Name	Birth Year	Gender
0	Oscar	1959	M
1	André	1976	M
2	Ariane	1965	F
3	Laure	1976	F
4	Max	1947	M

FIG. 1POLICIES

Row-ID	Type	Date	Client Link
0	Car	1998 06 24	2
1	Car	1998 09 09	1
2	House	1998 12 11	4
3	Car	1999 01 10	4
4	House	1999 01 31	2
5	Car	1999 02 01	3
6	House	1999 02 01	3

FIG. 2ACCIDENTS

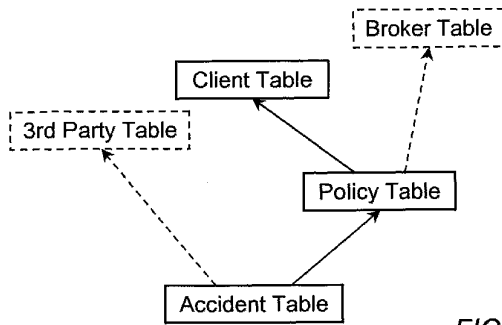
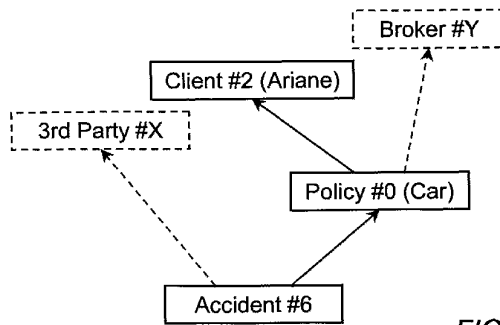
Row-ID	Date	Amount	Policy Link
0	1998 10 03	1 000	1
1	1999 04 18	10 000	4
2	1999 04 18	800	3
3	1999 06 12	1 300	1
4	1999 08 31	300	3
5	1999 12 08	2 200	5
6	1999 12 09	1 000	0
7	2000 02 25	6 000	4
8	2000 03 20	800	1

FIG. 3

WO 02/44943

PCT/IB01/02792

2/57

FIG. 4FIG. 5

WO 02/44943

PCT/IB01/02792

3/57

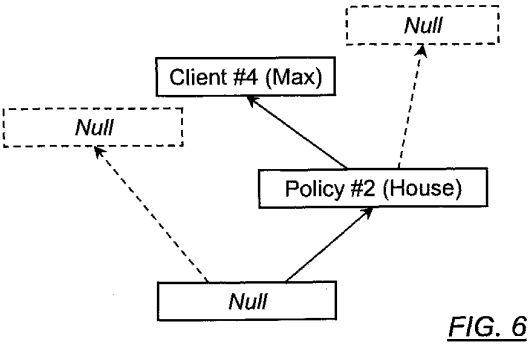


FIG. 6

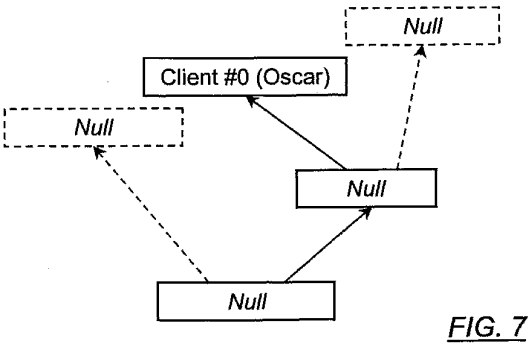


FIG. 7

FLAT FILE

Row-ID	Client Name	Birth Year	Gender	Policy Type	Policy Date	Acc. Date	Amount
0	André	1976	M	Car	1998 09 09	1998 10 03	1 000
1	Ariane	1965	F	House	1999 01 31	1999 04 18	10 000
2	Max	1947	M	Car	1999 01 10	1999 04 18	800
3	André	1976	M	Car	1998 09 09	1999 06 12	1 300
4	Max	1947	M	Car	1999 01 10	1999 08 31	300
5	Laure	1976	F	Car	1999 02 01	1999 12 08	2 200
6	Ariane	1965	F	Car	1998 06 24	1999 12 09	1 000
7	Ariane	1965	F	House	1999 01 31	2000 02 25	6 000
8	André	1976	M	Car	1998 09 09	2000 03 20	800
9	Max	1947	M	House	1998 12 11	Null	Null
10	Laure	1976	F	House	1999 02 01	Null	Null
11	Oscar	1959	M	Null	Null	Null	Null

FIG. 8

WO 02/44943

PCT/IB01/02792

5/57

LINK TABLE

FF Row-ID	Client Link	Policy Link	Accident Link
0	1	1	0
1	2	4	1
2	4	3	2
3	1	1	3
4	4	3	4
5	3	5	5
6	2	0	6
7	2	4	7
8	1	1	8
9	4	2	-1
10	3	6	-1
11	0	-1	-1

FIG. 9ACCIDENT AMOUNT (TRUNCATION LENGTH 3) THESAURUS

Amount Prefix	FF Row-ID List	Bitmap
0	{2,4,8}	001010001000
1	{0,3,6}	100100100000
2	{5}	000001000000
6	{7}	000000010000
10	{1}	010000000000

FIG. 10H

WO 02/44943

PCT/IB01/02792

6/57

CLIENT NAME THESAURUS

	Name	FF Row-ID List	Bitmap
0	<i>Null</i>	-	000000000000
1	André	{0,3,8}	100100001000
2	Ariane	{1,6,7}	010000110000
3	Laure	{5,10}	000001000010
4	Max	{2,4,9}	001010000100
5	Oscar	{11}	000000000001

FIG. 10ACLIENT BIRTH YEAR THESAURUS

	Birth Year	FF Row-ID List	Bitmap
0	<i>Null</i>	-	000000000000
1	1947	{2,4,9}	001010000100
2	1959	{11}	000000000001
3	1965	{1,6,7}	010000110000
4	1976	{0,3,5,8,10}	100101001010

FIG. 10BCLIENT GENDER THESAURUS

	Gender	FF Row-ID List	Bitmap
0	<i>Null</i>	-	000000000000
1	F	{1,5,6,7,10}	010001110010
2	M	{0,2,3,4,8,9,11}	101110001101

FIG. 10C

WO 02/44943

PCT/IB01/02792

7/57

POLICY TYPE THESAURUS

	Type	FF Row-ID List	Bitmap
0	<i>Null</i>	{11}	000000000001
1	Car	{0,2,3,4,5,6,8}	101111101000
2	House	{1,7,9,10}	010000010110

FIG. 10DPOLICY DATE THESAURUS

	Date	FF Row-ID List	Bitmap
0	<i>Null</i>	{11}	000000000001
1	1998 06 24	{6}	000000100000
2	1998 09 09	{0,3,8}	100100001000
3	1998 12 11	{9}	000000000100
4	1999 01 10	{2,4}	001010000000
5	1999 01 31	{1,7}	010000010000
6	1999 02 01	{5,10}	000001000010

FIG. 10E

ACCIDENT DATE THESAURUS

	Date	FF Row-ID List	Bitmap
0	<i>Null</i>	{9,10,11}	000000000111
1	1998 10 03	{0}	100000000000
2	1999 04 18	{1,2}	010000000000
3	1999 06 12	{3}	000100000000
4	1999 08 31	{4}	000010000000
5	1999 12 08	{5}	000001000000
6	1999 12 09	{6}	000000100000
7	2000 02 25	{7}	000000010000
8	2000 03 20	{8}	000000001000

FIG. 10FACCIDENT AMOUNT THESAURUS

	Amount	FF Row-ID List	Bitmap
0	<i>Null</i>	{9,10,11}	000000000111
1	300	{4}	000010000000
2	800	{2,8}	001000001000
3	1 000	{0,6}	100000100000
4	1 300	{3}	000100000000
5	2 200	{5}	000001000000
6	6 000	{7}	000000010000
7	10 000	{1}	010000000000

FIG. 10G

WO 02/44943

PCT/IB01/02792

9/57

CLIENT NAME THESAURUS

Name	Bitmap	L1 Ranks	L1 Segments
Null	000 000 000 000	-	-
André	100 100 001 000	{0,1,2}	{100,100,001}
Ariane	010 000 110 000	{0,2}	{010,110}
Laure	000 001 000 010	{1,3}	{001,010}
Max	001 010 000 100	{0,2,3}	{001,010,100}
Oscar	000 000 000 001	{3}	{001}

FIG. 11AACCIDENT AMOUNT THESAURUS

Amount	Bitmap	L1 Ranks	L1 Segments
Null	000 000 000 111	{3}	{111}
300	000 010 000 000	{1}	{010}
800	001 000 001 000	{0,2}	{001,001}
1 000	100 000 100 000	{0,2}	{100,100}
1 300	000 100 000 000	{1}	{100}
2 200	000 001 000 000	{1}	{001}
6 000	000 000 010 000	{2}	{010}
10 000	010 000 000 000	{0}	{010}

FIG. 11GACCIDENT AMOUNT MACROWORD (/ 3) THESAURUS

Prefix	Bitmap	L1 Ranks	L1 Segments
0	001 010 001 000	{0,1,2}	{001,010,001}
1	100 100 100 000	{0,1,2}	{100,100,100}
2	000 001 000 000	{1}	{001}
6	000 000 010 000	{2}	{010}
10	010 000 000 000	{0}	{010}

FIG. 11H

WO 02/44943

PCT/IB01/02792

10/57

CLIENT NAME THESAURUS

Name	L1 Ranks	L1 Bitmap	L2 Ranks	L2 Segments
<i>Null</i>	-	00 00	-	-
André	{0,1,2}	11 10	{0,1}	{11,10}
Ariane	{0,2}	10 10	{0,1}	{10,10}
Laure	{1,3}	01 01	{0,1}	{01,01}
Max	{0,2,3}	10 11	{0,1}	{10,11}
Oscar	{3}	00 01	{1}	{01}

FIG. 12AACCIDENT AMOUNT THESAURUS

Amount	L1 Ranks	L1 Bitmap	L2 Ranks	L2 Segments
<i>Null</i>	{3}	00 01	{1}	{01}
300	{1}	01 00	{0}	{01}
800	{0,2}	10 10	{0,1}	{10,10}
1 000	{0,2}	10 10	{0,1}	{10,10}
1 300	{1}	01 00	{0}	{01}
2 200	{1}	01 00	{0}	{01}
6 000	{2}	00 10	{1}	{10}
10 000	{0}	10 00	{0}	{10}

FIG. 12GACCIDENT AMOUNT MACROWORD (/ 3) THESAURUS

Prefix	L1 Ranks	L1 Bitmap	L2 Ranks	L2 Segments
0	{0,1,2}	11 10	{0,1}	{11,10}
1	{0,1,2}	11 10	{0,1}	{11,10}
2	{1}	01 00	{0}	{01}
6	{2}	00 10	{1}	{10}
10	{0}	10 00	{0}	{10}

FIG. 12H

WO 02/44943

PCT/IB01/02792

11/57

CLIENT NAME THESAURUS

Name	L1 Ranks	L1 Segments	L2 Ranks	L2 Segments
<i>Null</i>	-	-	-	-
André	{0,1,2}	{100,100,001}	{0,1}	{11,10}
Ariane	{0,2}	{010,110}	{0,1}	{10,10}
Laure	{1,3}	{001,010}	{0,1}	{01,01}
Max	{0,2,3}	{001,010,100}	{0,1}	{10,11}
Oscar	{3}	{001}	{1}	{01}

FIG. 13AACCIDENT AMOUNT THESAURUS

Amount	L1 Ranks	L1 Segments	L2 Ranks	L2 Segments
<i>Null</i>	{3}	{111}	{1}	{01}
300	{1}	{010}	{0}	{01}
800	{0,2}	{001,001}	{0,1}	{10,10}
1 000	{0,2}	{100,100}	{0,1}	{10,10}
1 300	{1}	{100}	{0}	{01}
2 200	{1}	{001}	{0}	{01}
6 000	{2}	{010}	{1}	{10}
10 000	{0}	{010}	{0}	{10}

FIG. 13GACCIDENT AMOUNT MACROWORD (/ 3) THESAURUS

Prefix	L1 Ranks	L1 Segments	L2 Ranks	L2 Segments
0	{0,1,2}	{001,010,001}	{0,1}	{11,10}
1	{0,1,2}	{100,100,100}	{0,1}	{11,10}
2	{1}	{001}	{0}	{01}
6	{2}	{010}	{1}	{10}
10	{0}	{010}	{0}	{10}

FIG. 13H

WO 02/44943

PCT/IB01/02792

12/57

CLIENT NAME THESAURUS

Name	L1 First	L1 Last	L2 First
<i>Null</i>	0	0	0
André	20	2	3
Ariane	5	6	12
Laure	8	12	14
Max	29	15	6
Oscar	26	26	7

FIG. 14AACCIDENT AMOUNT THESAURUS

Amount	L1 First	L1 Last	L2 First
<i>Null</i>	13	13	21
300	4	4	22
800	9	10	16
1 000	3	19	18
1 300	24	24	20
2 200	25	25	4
6 000	23	23	9
10 000	16	16	10

FIG. 14GACCIDENT AMOUNT MACROWORD (/ 3) THESAURUS

Prefix	L1 First	L1 Last	L2 First
0	21	7	19
1	31	14	25
2	1	1	27
6	28	28	28
10	18	18	1

FIG. 14H

WO 02/44943

PCT/IB01/02792

13/57

Address L1 Rank Next L1 Segment

1	1	0	001
2	2	0	001
3	0	19	100
4	1	0	010
5	0	6	010
6	2	0	110
7	2	0	001
8	1	12	001
9	0	10	001
10	2	0	001
11	1	2	100
12	3	0	010
13	3	0	111
14	2	0	100
15	3	0	100
16	0	0	010
17	x	0	000
18	0	0	010
19	2	0	100
20	0	11	100
21	0	22	001
22	1	7	010
23	2	0	010
24	1	0	100
25	1	0	001
26	3	0	001
27	2	15	010
28	2	0	010
29	0	27	001
30	1	14	100
31	0	30	100
32	x	33	000
33	x	17	000

FIG. 15

WO 02/44943

PCT/IB01/02792

14/57

Address	L2 Rank	Next	L2 Segment
1	0	0	10
2	1	0	01
3	0	11	11
4	0	0	01
5	1	0	10
6	0	13	10
7	1	0	01
8	x	17	00
9	1	0	10
10	0	0	10
11	1	0	10
12	0	23	10
13	1	0	11
14	0	2	01
15	1	0	10
16	0	15	10
17	x	24	00
18	0	5	10
19	0	30	11
20	0	0	01
21	1	0	01
22	0	0	01
23	1	0	10
24	x	0	00
25	0	26	11
26	1	0	10
27	0	0	01
28	1	0	10
29	x	8	00
30	1	0	10

FIG. 16

WO 02/44943

PCT/IB01/02792

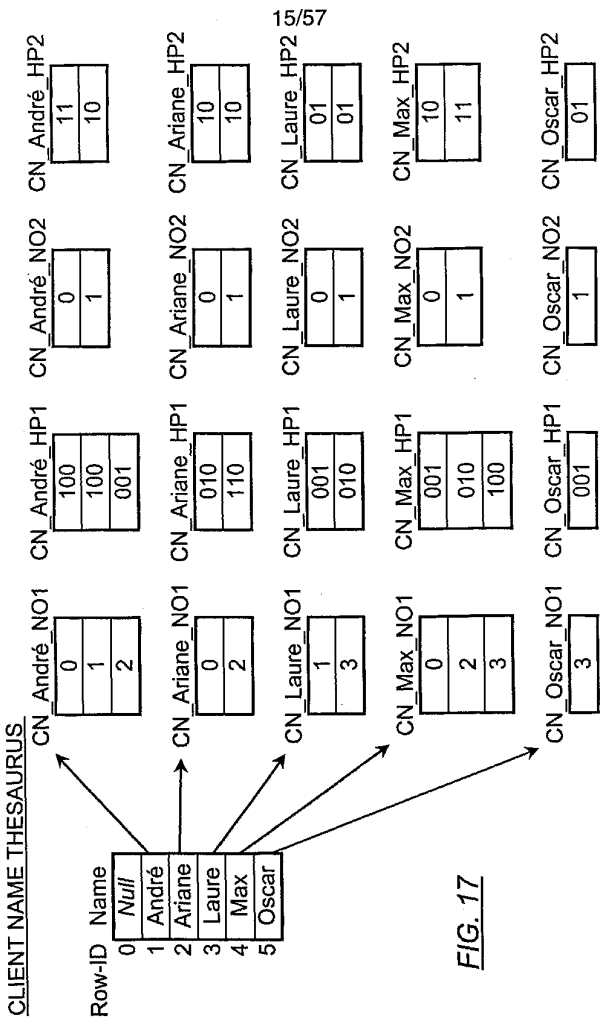
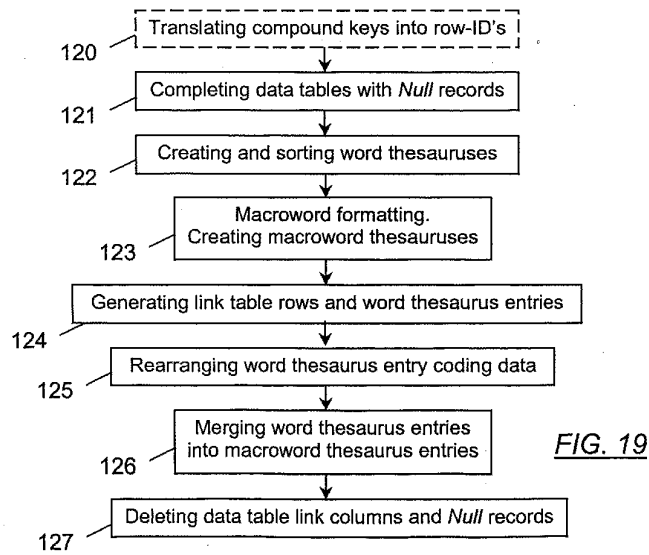
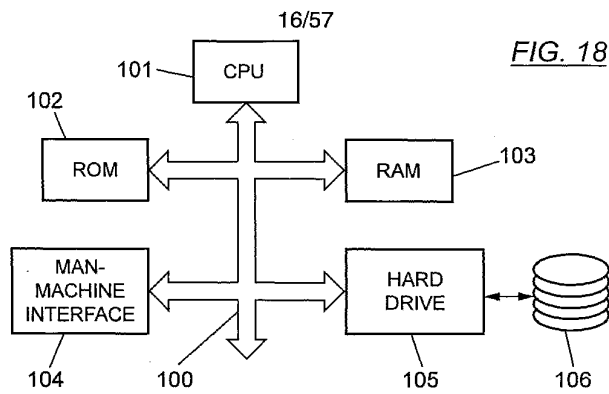


FIG. 17

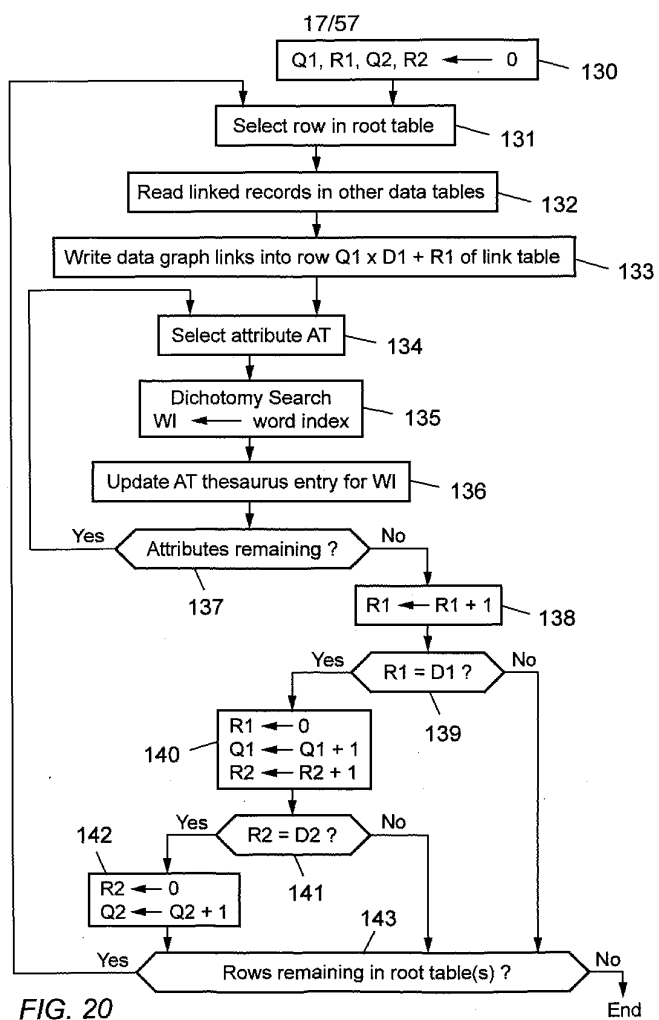
WO 02/44943

PCT/IB01/02792



WO 02/44943

PCT/IB01/02792



WO 02/44943

PCT/IB01/02792

18/57

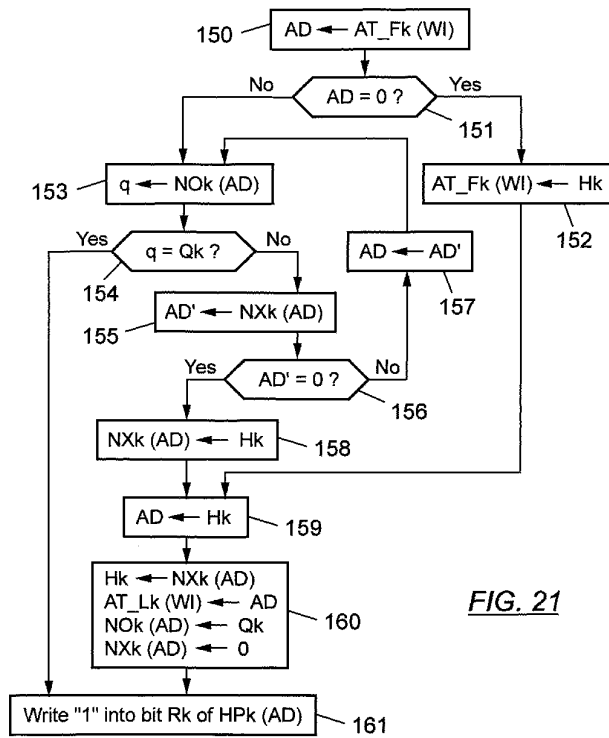
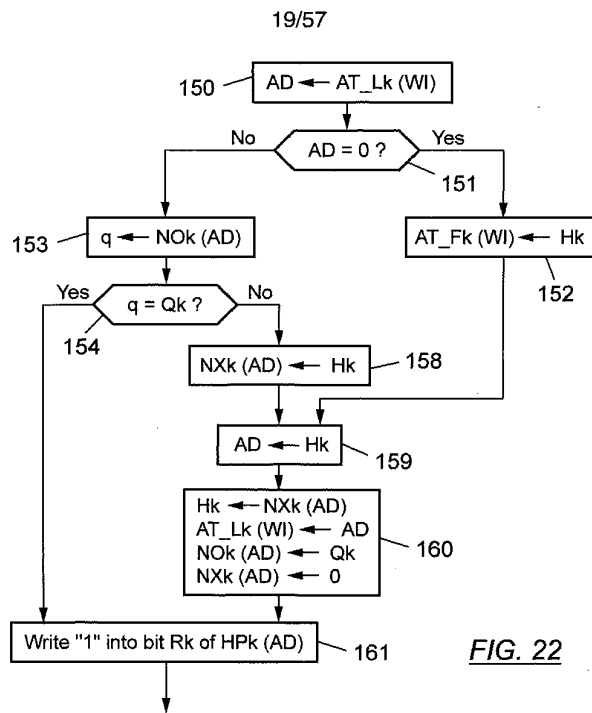


FIG. 21

WO 02/44943

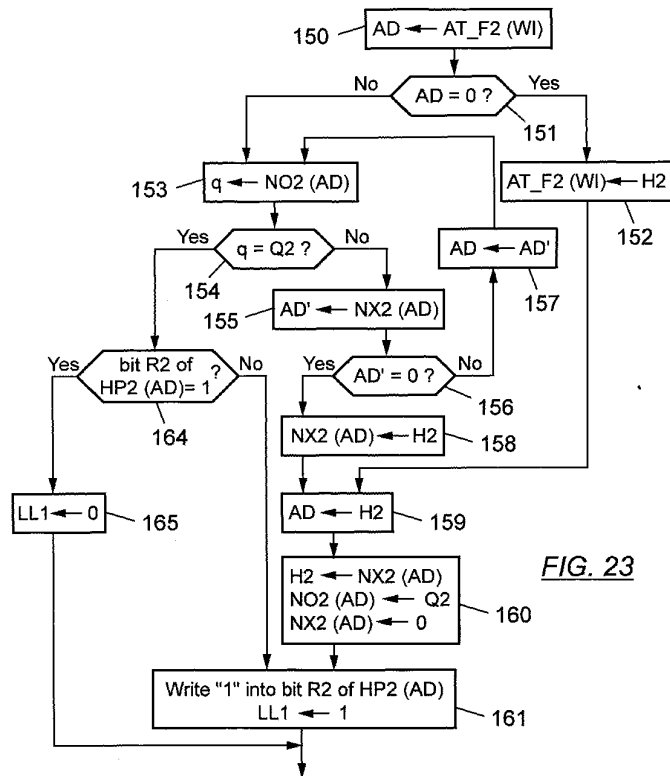
PCT/IB01/02792



WO 02/44943

PCT/IB01/02792

20/57



WO 02/44943

PCT/IB01/02792

21/57

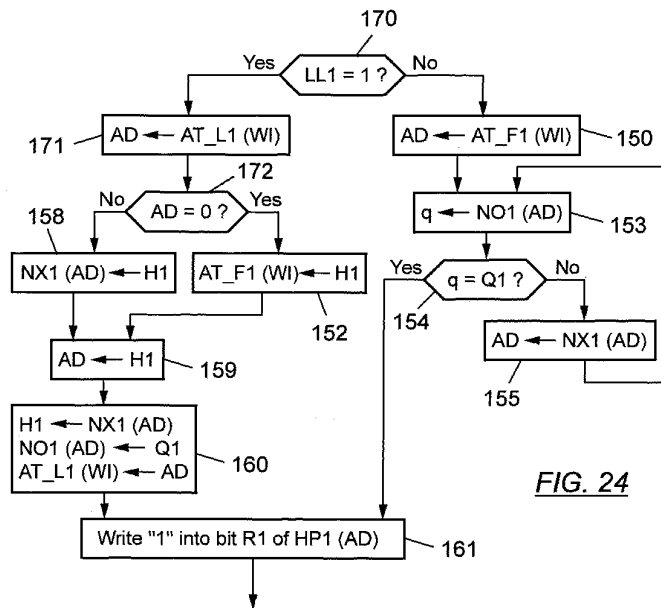


FIG. 24

WO 02/44943

PCT/IB01/02792

22/57

CLIENT NAME THESAURUS

Row-ID	Name	Word Index	Format
0	<i>Null</i>	0	1
1	André	1	1
2	Ariane	2	1
3	Laure	3	1
4	Max	4	1
5	Oscar	5	1

FIG. 25ACCIDENT AMOUNT THESAURUS

Row-ID	Amount	Word Index	Format
0	<i>Null</i>	0	1
1	300	1	1
2	800	2	1
3	1 000	3	1
4	1 300	4	1
5	2 200	5	1
6	6 000	6	1
7	10 000	7	1

FIG. 26ACC. AMOUNT. MACROWORD (/ 3) THESAURUS

Row-ID	Prefix	Macro Index	First Word	Format
0	0	0	1	1
1	1	1	3	1
2	2	2	5	1
3	6	3	6	1
4	10	4	7	1

FIG. 27

WO 02/44943

PCT/IB01/02792

23/57

CLIENT NAME AUXILIARY TABLE

Row-ID	CN F1	CN L1	CN F2
0	0	0	0
1	1	3	1
2	4	5	3
3	6	7	5
4	8	10	7
5	11	11	9

FIG. 28ACCIDENT AMOUNT AUXILIARY TABLE

Row-ID	AA F1	AA L1	AA F2
0	12	12	10
1	13	13	11
2	14	15	12
3	16	17	14
4	18	18	16
5	19	19	17
6	20	20	18
7	21	21	19

FIG. 29ACC. AMOUNT. MACROWORD (/ 3) AUXILIARY TABLE

Row-ID	AA 3 F1	AA 3 L1	AA 3 F2
0	22	24	20
1	25	27	22
2	28	28	24
3	29	29	25
4	30	30	26

FIG. 30

WO 02/44943

PCT/IB01/02792

24/57

AD	NO1	NX1	HP1
1	0	2	100
2	1	3	100
3	2	0	001
4	0	5	010
5	2	0	110
6	1	7	001
7	3	0	010
8	0	9	001
9	2	10	010
10	3	0	100
11	3	0	001
12	3	0	111
13	1	0	010
14	0	15	001
15	2	0	001
16	0	17	100
17	2	0	100
18	1	0	100
19	1	0	001
20	2	0	010
21	0	0	010
22	0	23	001
23	1	24	010
24	2	0	001
25	0	26	100
26	1	27	100
27	2	0	100
28	1	0	001
29	2	0	010
30	0	0	010
31	x	32	000
32	x	33	000
33	x	0	000

FIG. 31

WO 02/44943

PCT/IB01/02792

25/57

AD	NO2	NX2	HP2
1	0	2	11
2	1	0	10
3	0	4	10
4	1	0	10
5	0	6	01
6	1	0	01
7	0	8	10
8	1	0	11
9	1	0	01
10	1	0	01
11	0	0	01
12	0	13	10
13	1	0	10
14	0	15	10
15	1	16	10
16	0	17	01
17	0	18	01
18	1	19	10
19	0	20	10
20	0	21	11
21	1	0	10
22	0	23	11
23	1	0	10
24	0	0	01
25	1	0	10
26	0	0	10
27	x	28	00
28	x	29	00
29	x	30	00
30	x	0	00

FIG. 32

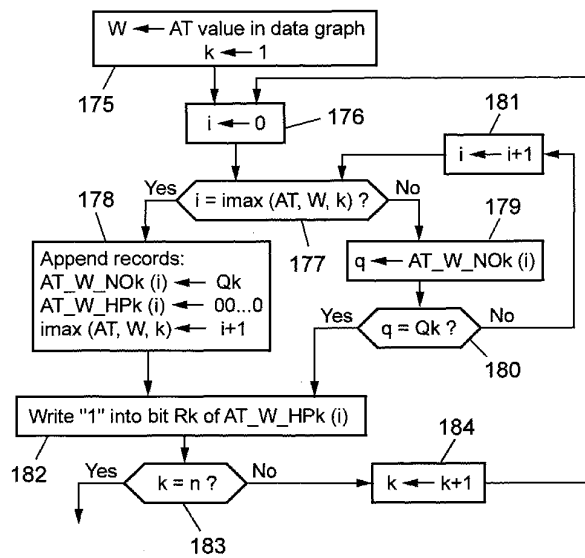


FIG. 33

WO 02/44943

PCT/IB01/02792

27/57

AD	NO2	NX2		HP2		F1
1	0	2		11		1
2	1	0		10		3
3	0	4		10		4
4	1	0		10		5
5	0	6		01		6
6	1	0		01		7
7	0	8		10		8
8	1	0		11		9
9	1	0		01		11
10	1	0		01		12
11	0	0		01		13
12	0	13		10		14
13	1	0		10		15
14	0	15		10		16
15	1	16		10		17
16	0	17		01		18
17	0	18		01		19
18	1	19		10		20
19	0	20		10		21
20	0	21		11		22
21	1	0		10		24
22	0	23		11		25
23	1	0		10		27
24	0	0		01		28
25	1	0		10		29
26	0	0		10		30
27	x	28		00		x
28	x	29		00		x
29	x	30		00		x
30	x	0		00		x

FIG. 34A

WO 02/44943

PCT/IB01/02792

28/57

AD	NX1	HP1
1	2	100
2	0	100
3	0	001
4	0	010
5	0	110
6	0	001
7	0	010
8	0	001
9	10	010
10	0	100
11	0	001
12	0	111
13	0	010
14	0	001
15	0	001
16	0	100
17	0	100
18	0	100
19	0	001
20	0	010
21	0	010
22	23	001
23	0	010
24	0	001
25	26	100
26	0	100
27	0	100
28	0	001
29	0	010
30	0	010
31	32	000
32	33	000
33	0	000

FIG. 34B

WO 02/44943

PCT/IB01/02792

29/57

AD	NO2	NX2		F1
1	0	2		1
2	1	0		3
3	0	4		4
4	1	0		5
5	0	6		6
6	1	0		7
7	0	8		8
8	1	0		9
9	1	0		11
10	1	0		12
11	0	0		13
12	0	13		10
13	1	0		14
14	0	15		15
15	1	16		16
16	0	17		17
17	0	18		18
18	1	19		19
19	0	20		20
20	0	21		21
21	1	0		22
22	0	23		24
23	1	0		25
24	0	0		27
25	1	0		28
26	0	0		29
27	x	28		30
28	x	29		x
29	x	30		x
30	x	0		x

FIG. 34C

WO 02/44943

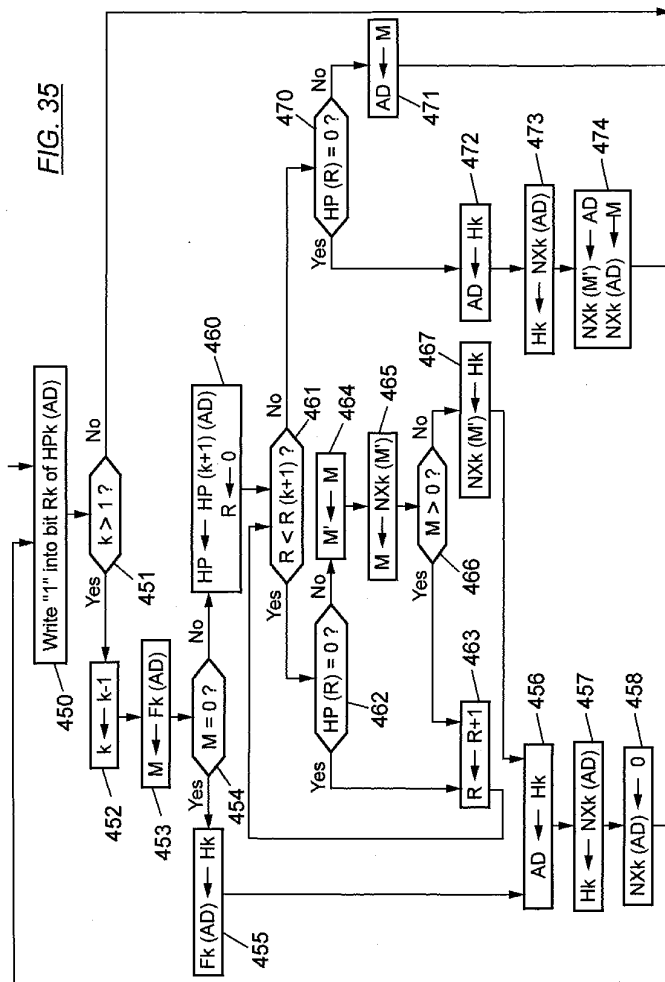
PCT/IB01/02792

30/57

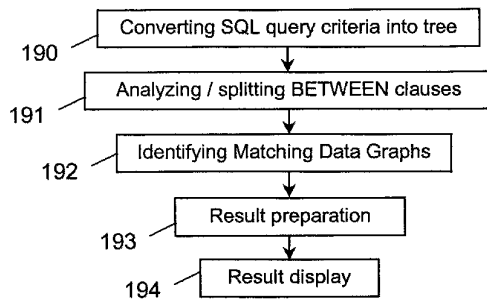
AD	R2	NX1	HP1
1	0	2	100
2	1	0	100
3	0	0	001
4	0	0	010
5	0	0	110
6	1	0	001
7	1	0	010
8	0	0	001
9	0	10	010
10	1	0	100
11	1	0	001
12	1	0	111
13	1	0	010
14	0	0	001
15	0	0	001
16	0	0	100
17	0	0	100
18	1	0	100
19	1	0	001
20	0	0	010
21	0	0	010
22	0	23	001
23	1	0	010
24	0	0	001
25	0	26	100
26	1	0	100
27	0	0	100
28	1	0	001
29	0	0	010
30	0	0	010
31	x	32	000
32	x	33	000
33	x	0	000

FIG. 34D

31/57



32/57

FIG. 36

WO 02/44943

PCT/IB01/02792

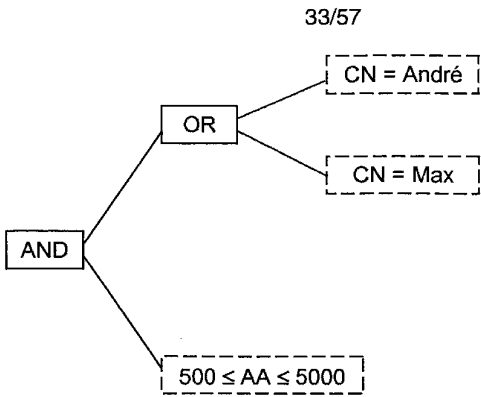


FIG. 37

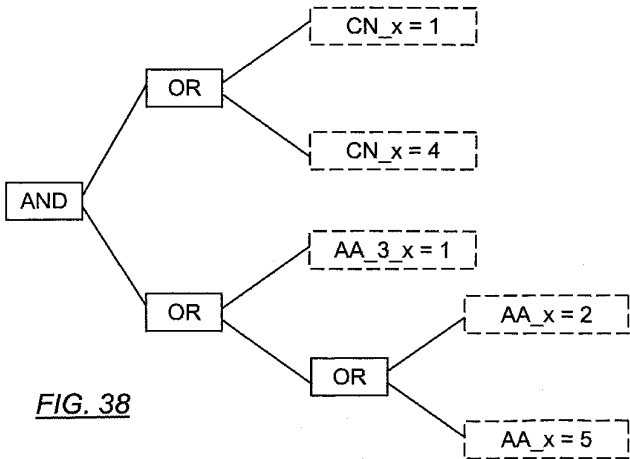


FIG. 38

WO 02/44943

PCT/IB01/02792

34/57

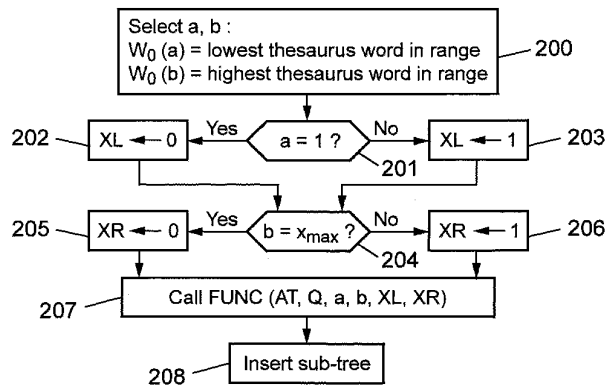
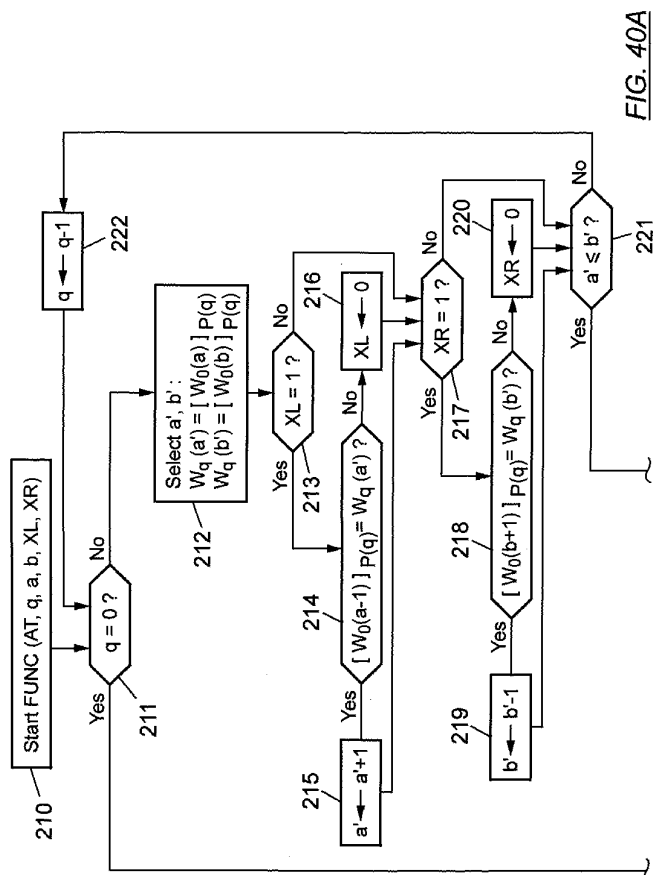


FIG. 39

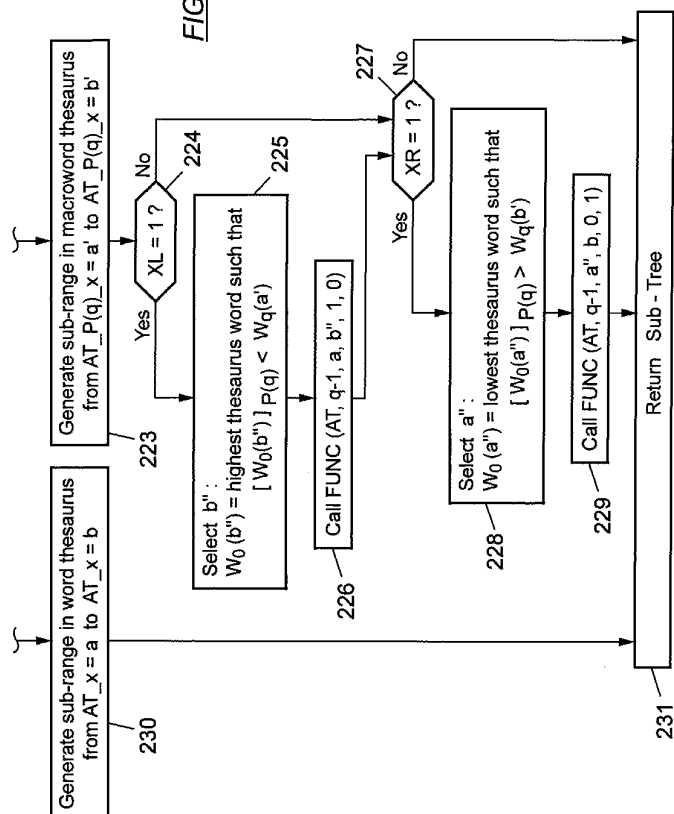


WO 02/44943

PCT/IB01/02792

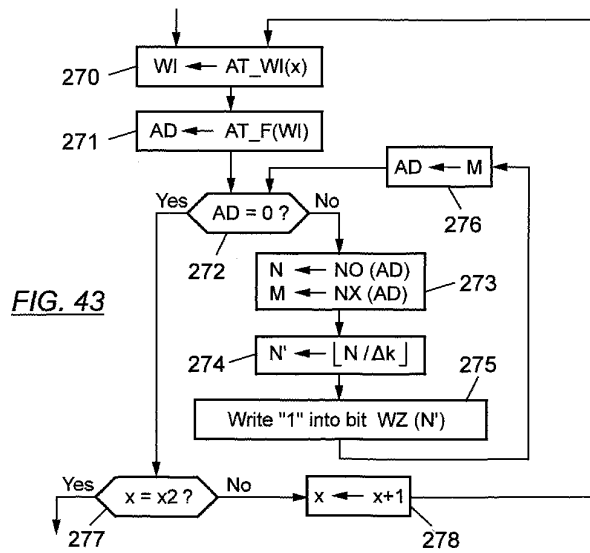
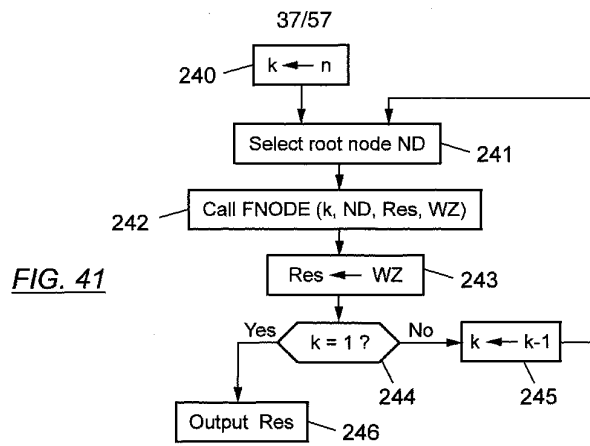
36/57

FIG. 40B



WO 02/44943

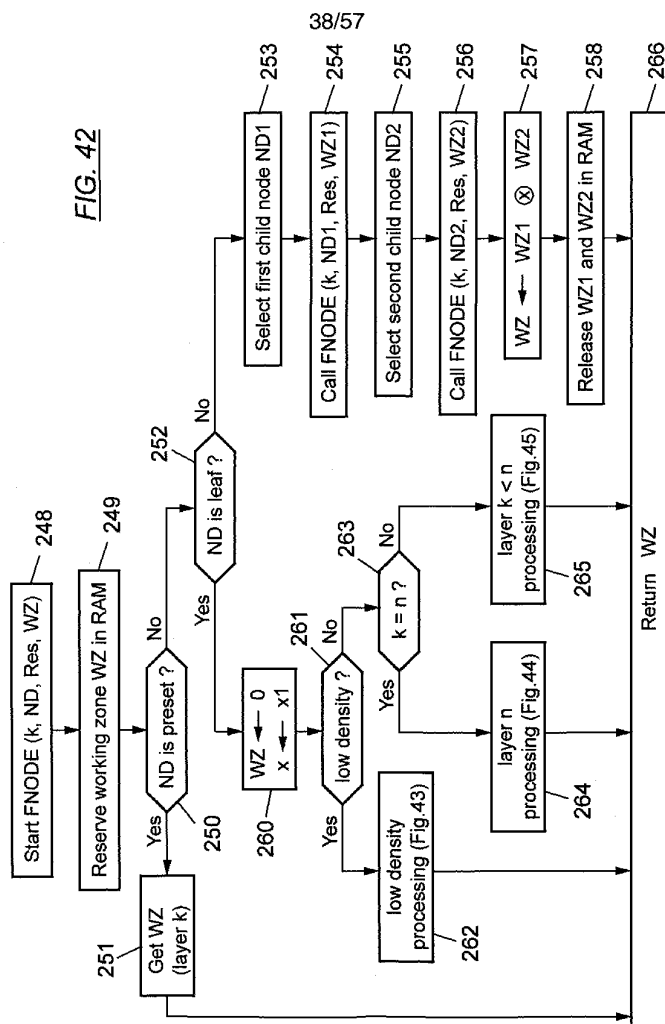
PCT/IB01/02792



WO 02/44943

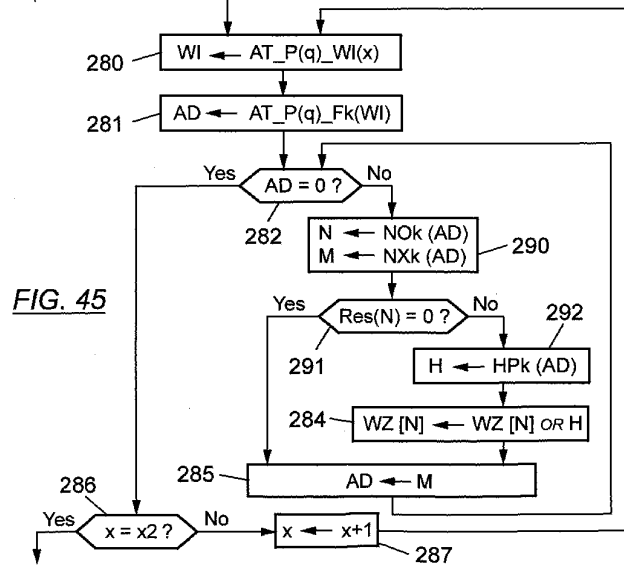
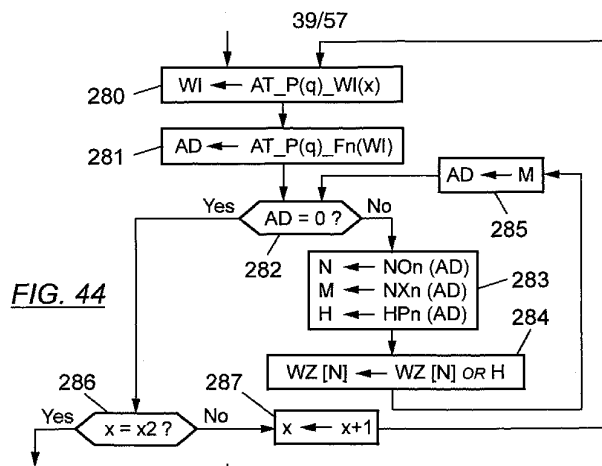
PCT/IB01/02792

FIG. 42



WO 02/44943

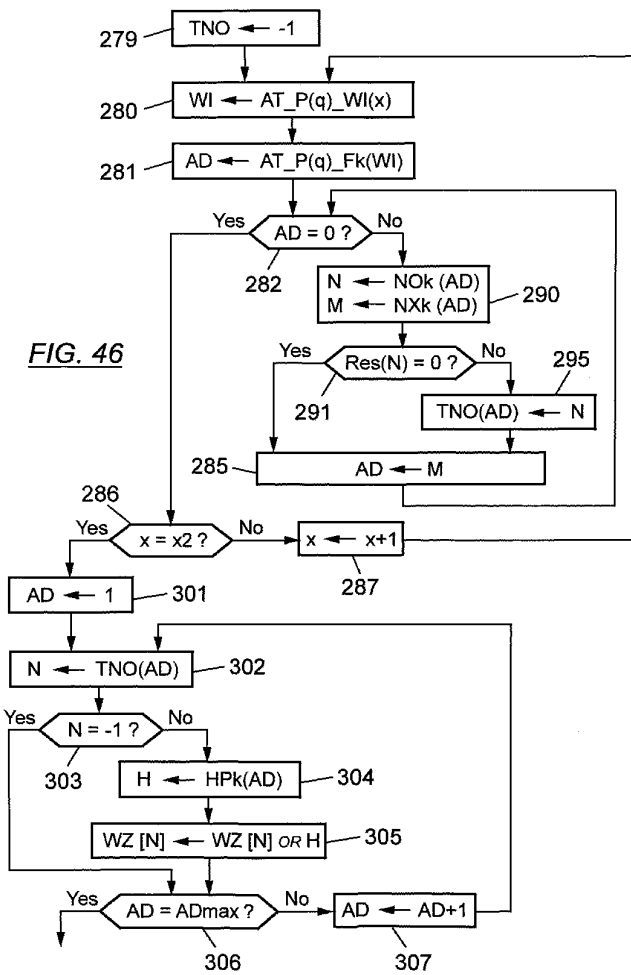
PCT/IB01/02792



WO 02/44943

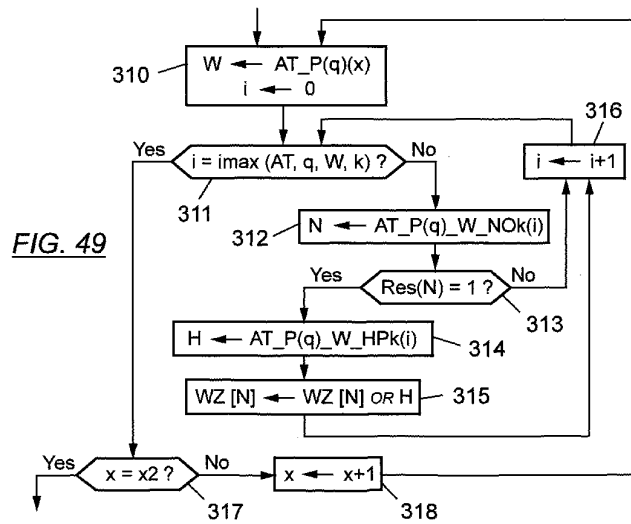
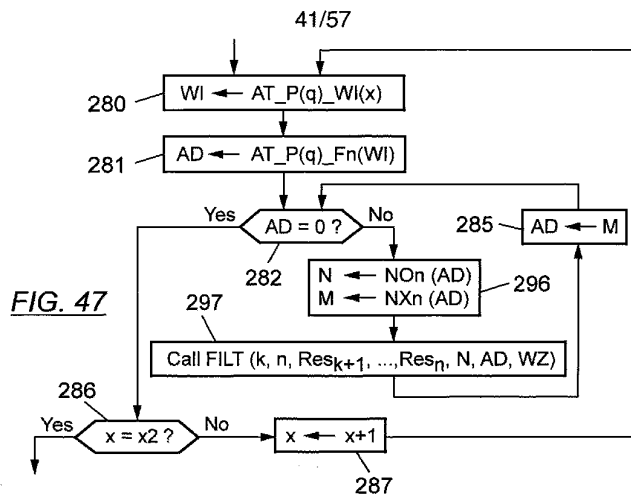
PCT/IB01/02792

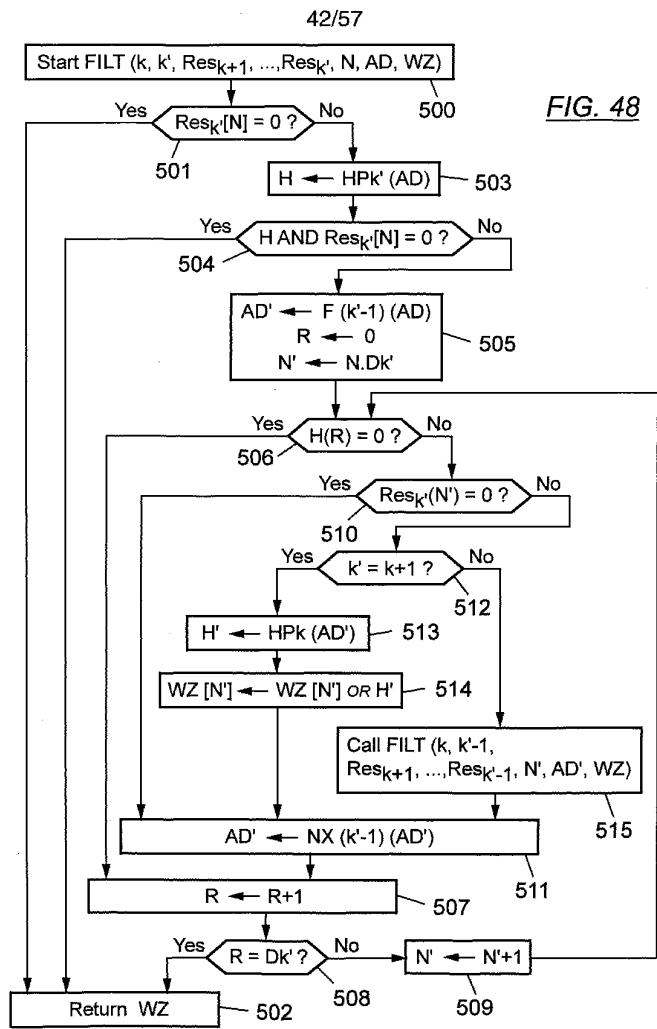
40/57



WO 02/44943

PCT/IB01/02792





WO 02/44943

PCT/IB01/02792

43/57

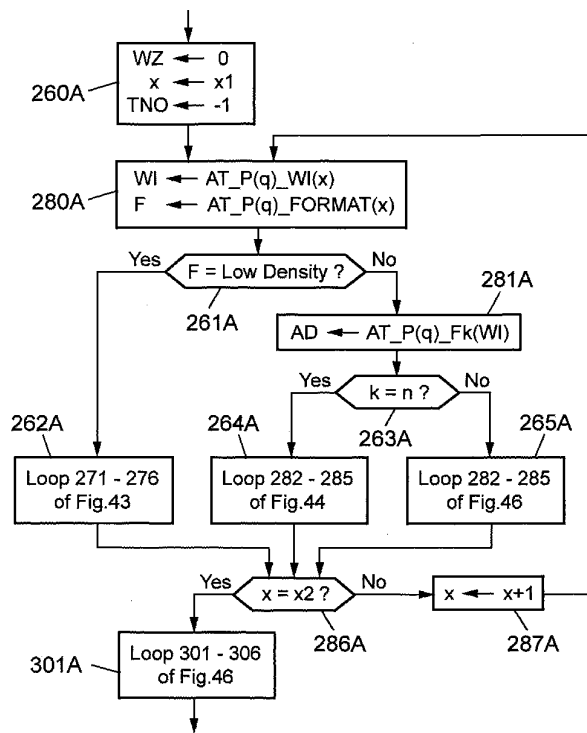


FIG. 50

WO 02/44943

PCT/IB01/02792

44/57

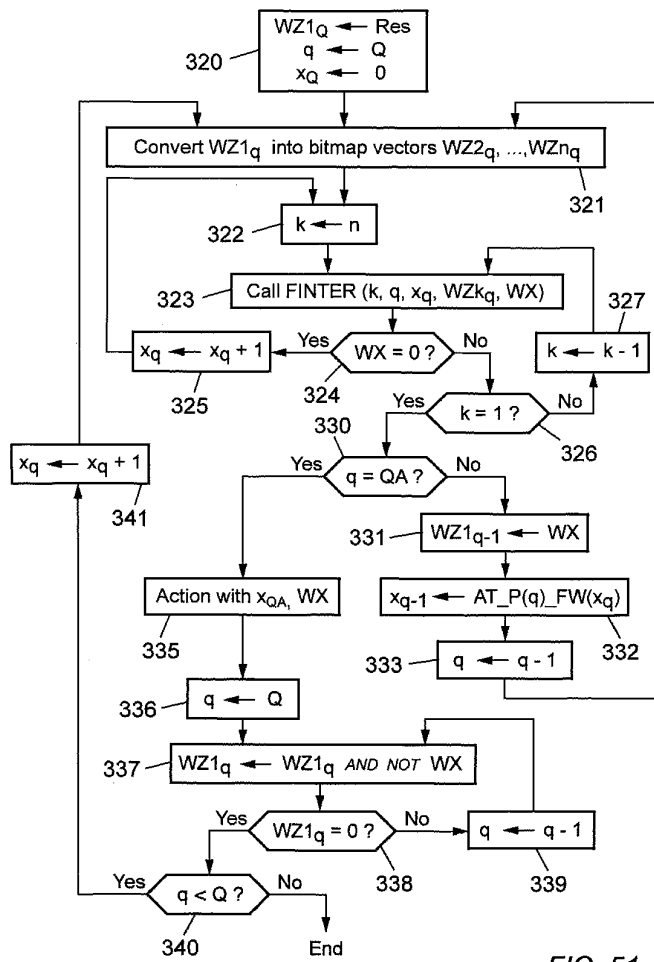
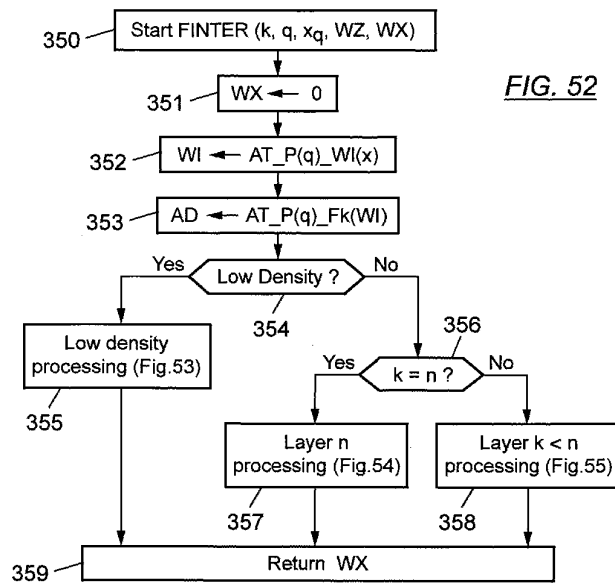


FIG. 51



WO 02/44943

PCT/IB01/02792

46/57

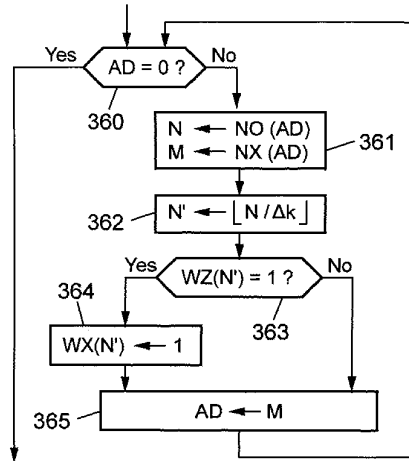


FIG. 53

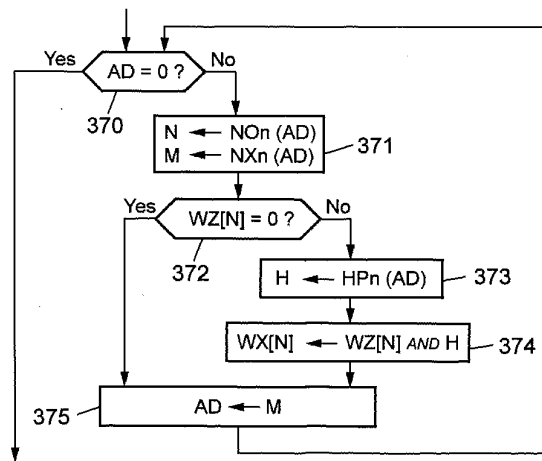


FIG. 54

WO 02/44943

PCT/IB01/02792

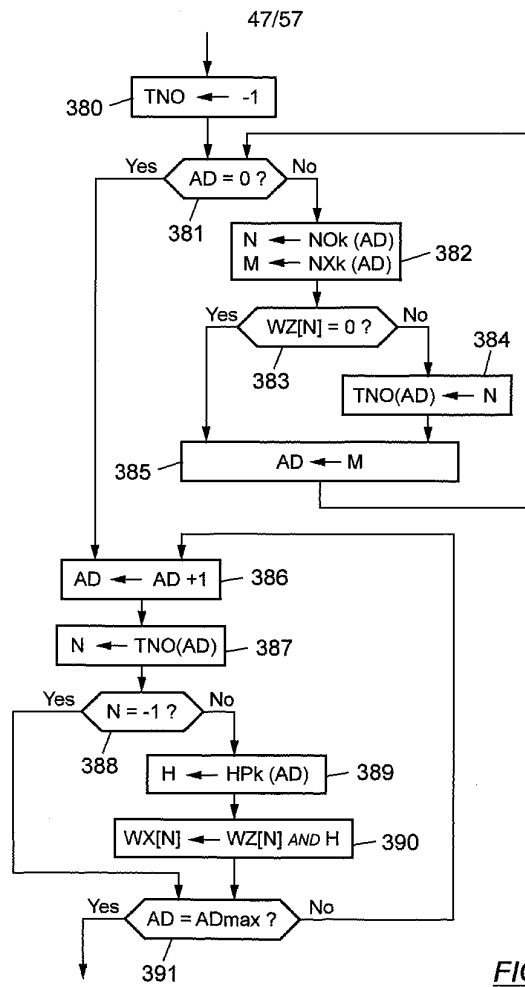


FIG. 55

WO 02/44943

PCT/IB01/02792

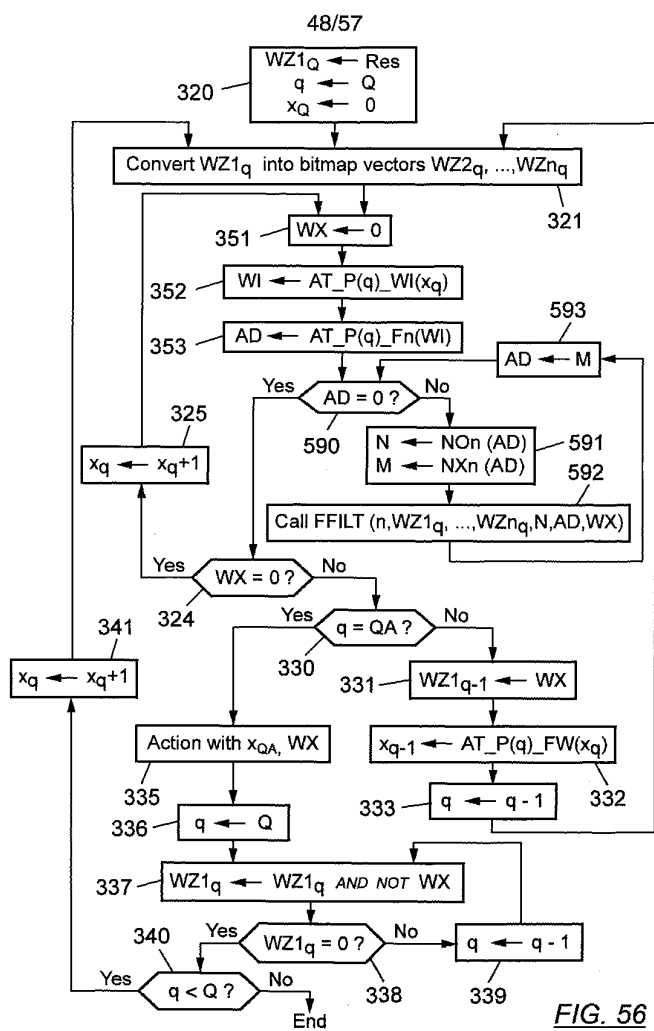
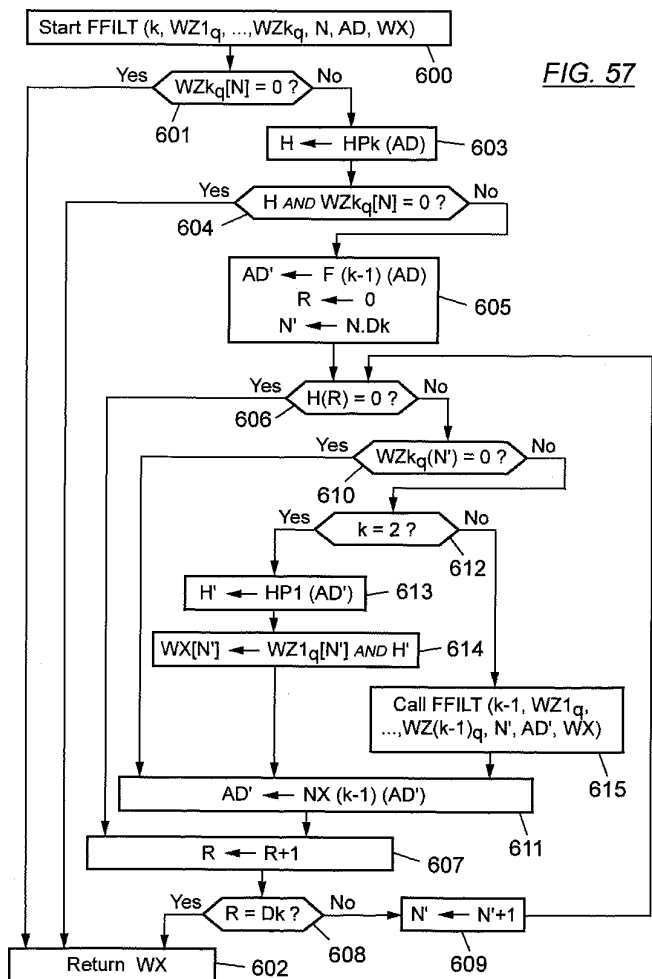


FIG. 56

WO 02/44943

PCT/IB01/02792

49/57



WO 02/44943

PCT/IB01/02792

50/57

CLIENT NAME

NO2 F AD'

0	1
1	5

FIG. 58

ACC. AMOUNT

NO2 F AD'

0	10
1	16

FIG. 59

ACC. AMOUNT / 3

NO2 F AD'

0	20
1	24

FIG. 60

AD'	HP'2	NX'2	PTR
1	11	2	1
2	10	3	2
3	01	4	3
4	10	0	4
5	10	6	1
6	10	7	2
7	01	8	3
8	11	9	4
9	01	0	5
10	01	11	1
11	10	12	2
12	10	13	3
13	01	14	4
14	01	15	5
15	10	0	7
16	01	17	0
17	10	18	2
18	10	19	3
19	10	0	6
20	11	21	0
21	11	22	1
22	01	23	2
23	10	0	4
24	10	25	0
25	10	26	1
26	10	0	3
27	00	28	x
28	00	29	x
29	00	30	x
30	00	0	x

FIG. 61

WO 02/44943

PCT/IB01/02792

51/57

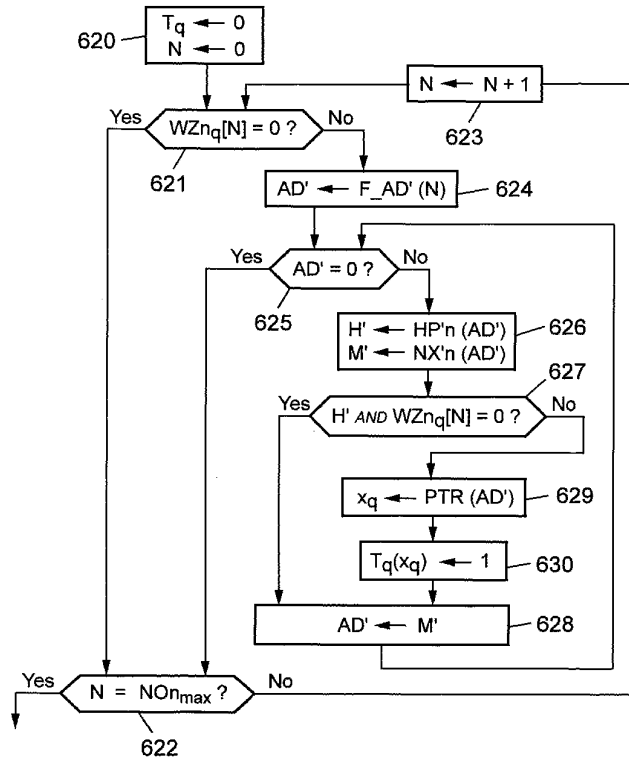


FIG. 62

WO 02/44943

PCT/IB01/02792

52/57

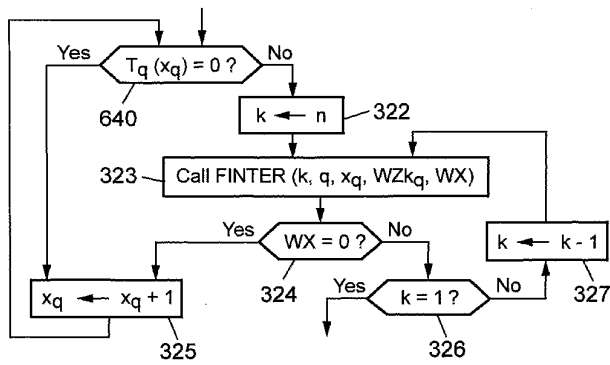


FIG. 63

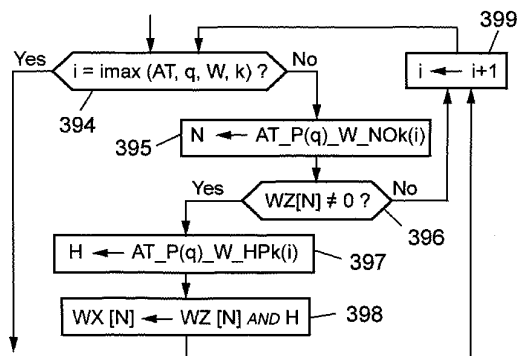


FIG. 64

WO 02/44943

PCT/IB01/02792

53/57

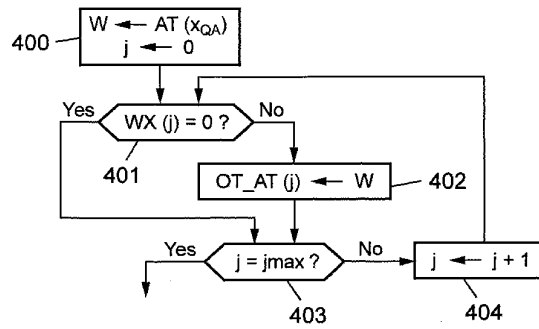


FIG. 65

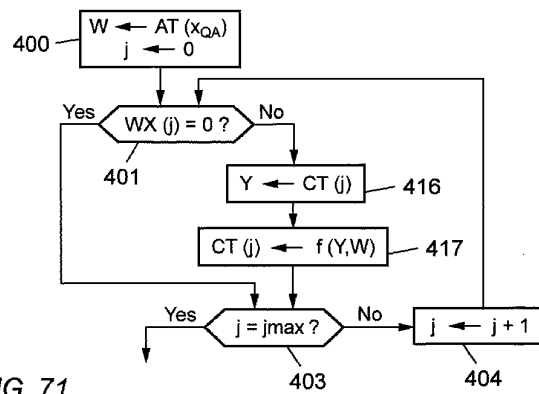


FIG. 71

WO 02/44943

PCT/IB01/02792

54/57

OUTPUT TABLE

Flat File Row-ID	Acc. Date	Client Name	Policy Date
0	1998 10 03	André	1998 09 09
1	/	/	/
2	1999 04 18	Max	1999 01 10
3	1999 06 12	André	1998 09 09
4	/	/	/
5	/	/	/
6	/	/	/
7	/	/	/
8	2000 03 20	André	1998 09 09
9	/	/	/
10	/	/	/
11	/	/	/

FIG. 66OUTPUT TABLE

Flat File Row-ID	Acc. Date	Client Name	Policy Date
0	1	1	2
1	/	/	/
2	2	4	4
3	3	1	2
4	/	/	/
5	/	/	/
6	/	/	/
7	/	/	/
8	8	1	2
9	/	/	/
10	/	/	/
11	/	/	/

FIG. 67

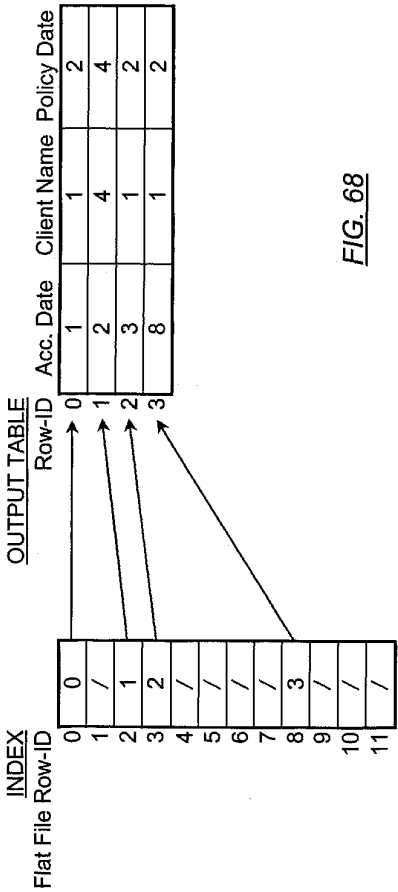
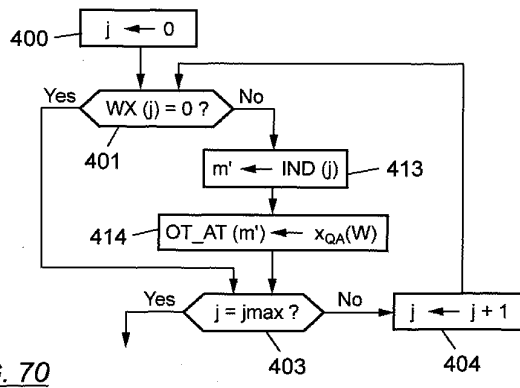
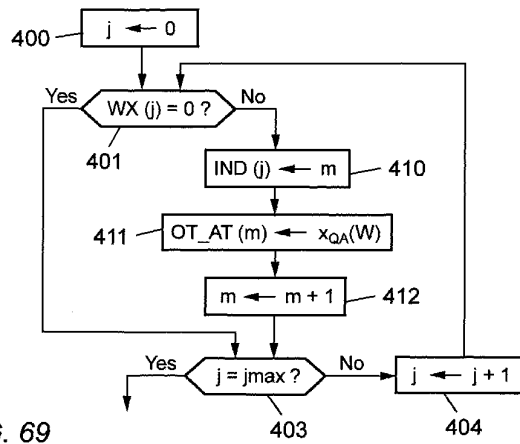


FIG. 68

WO 02/44943

PCT/IB01/02792

56/57



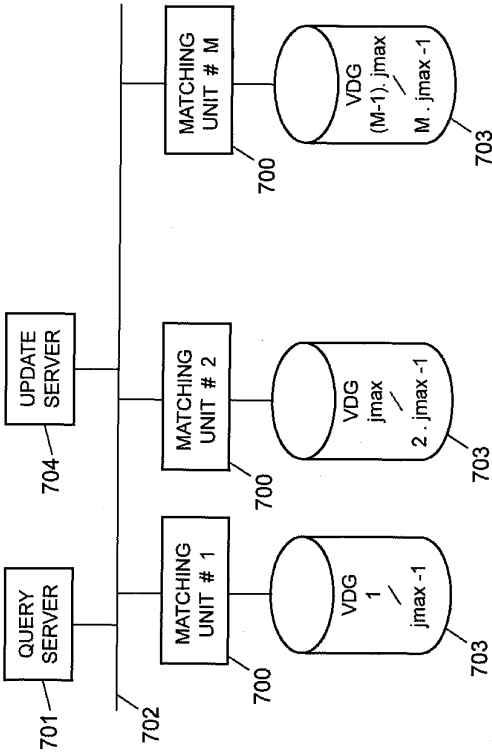


FIG. 72

【国際公開パンフレット（コレクトバージョン）】

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

REVISED VERSION

(19) World Intellectual Property
Organization
International Bureau



(43) International Publication Date
6 June 2002 (06.06.2002)

PCT

(10) International Publication Number
WO 2002/044943 A2

(51) International Patent Classification⁷: G06F 17/30

(21) International Application Number:
PCT/IB2001/002792

(22) International Filing Date:
29 November 2001 (29.11.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
00403329.6 29 November 2000 (29.11.2000) EP
00403330.4 29 November 2000 (29.11.2000) EP
00403331.2 29 November 2000 (29.11.2000) EP
00403332.0 29 November 2000 (29.11.2000) EP

(71) Applicant: LAFAYETTE SOFTWARE INC. [US/US];
Five Palo Alto Square, 3000 El Camino Real, Palo Alto,
CA 94306 2155 (US).

(72) Inventor: KOSKAS, Elie, Ouzi; 9, Allée Eridan, F-95350
Saint Brice Sous Foret (FR).

(81) Designated States (national): AE, AG, AL, AM, AT, AU,
AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU,

CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH,
GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC,
LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW,
MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG,
SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU,
ZA, ZM, ZW.

(84) Designated States (regional): ARIPO patent (GH, GM,
KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW),
Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM),
European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR,
GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent
(BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR,
NE, SN, TD, TG).

Published:
— with declaration under Article 17(2)(a): without abstract;
title not checked by the International Searching Authority

(48) Date of publication of this revised version: 1 April 2004

(15) Information about Correction:
see PCT Gazette No. 14/2004 of 1 April 2004, Section II

For two-letter codes and other abbreviations, refer to the "Guide
Notes on Codes and Abbreviations" appearing at the begin-
ning of each regular issue of the PCT Gazette.



WO 2002/044943 A2

(54) Title: METHODS OF ORGANIZING DATA AND PROCESSING QUERIES IN A DATABASE SYSTEM, AND DATABASE
SYSTEM AND SOFTWARE PRODUCT FOR IMPLEMENTING SUCH METHODS

(57) Abstract:

PATENT COOPERATION TREATY

PCT

DECLARATION OF NON-ESTABLISHMENT OF INTERNATIONAL SEARCH REPORT

(PCT Article 17(2)(a), Rules 13ter.1(c) and Rule 39)

Applicant's or agent's file reference BCT010154	IMPORTANT DECLARATION	Date of mailing (day/month/year) 05/02/2004
International application No. PCT/IB 01/02792	International filing date (day/month/year) 29/11/2001	(Earliest) Priority date (day/month/year) 29/11/2000
International Patent Classification (IPC) or both national classification and IPC G06F17/30		
Applicant LAFAYETTE SOFTWARE INC.		

This International Searching Authority hereby declares, according to Article 17(2)(a), that **no international search report will be established** on the international application for the reasons indicated below

1. ☐ The subject matter of the international application relates to:

- a. ☐ scientific theories.
- b. ☐ mathematical theories
- c. ☐ plant varieties.
- d. ☐ animal varieties.
- e. ☐ essentially biological processes for the production of plants and animals, other than microbiological processes and the products of such processes.
- f. ☐ schemes, rules or methods of doing business.
- g. ☐ schemes, rules or methods of performing purely mental acts.
- h. ☐ schemes, rules or methods of playing games.
- i. ☐ methods for treatment of the human body by surgery or therapy.
- j. ☐ methods for treatment of the animal body by surgery or therapy.
- k. ☐ diagnostic methods practised on the human or animal body.
- l. ☐ mere presentations of information.
- m. ☐ computer programs for which this International Searching Authority is not equipped to search prior art.

2. ☒ The failure of the following parts of the international application to comply with prescribed requirements prevents a meaningful search from being carried out:

☐ the description ☒ the claims ☐ the drawings

3. ☐ The failure of the nucleotide and/or amino acid sequence listing to comply with the standard provided for in Annex C of the Administrative Instructions prevents a meaningful search from being carried out:

☐ the written form has not been furnished or does not comply with the standard.

☐ the computer readable form has not been furnished or does not comply with the standard.

4. Further comments:

PLEASE SEE "FURTHER INFORMATION CONTINUED FORM PCT/ISA/203"

Name and mailing address of the International Searching Authority  European Patent Office, P.B. 5818 Patentlaan 2 NL-2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Authorized officer Selwa Harris

International Application No. PCT/18 01/02792

FURTHER INFORMATION CONTINUED FROM PCT/ISA/ 203

In view of the large number and also the wording of the claims presently on file, which render it difficult, if not impossible, to determine the matter for which protection is sought, the present application fails to comply with the clarity and conciseness requirements of Article 6 PCT (see also Rule 6.1(a) PCT) to such an extent that a meaningful search on the basis of the claims is impossible. Consequently, no search report can be established for the present application.

The applicant's attention is drawn to the fact that claims relating to inventions in respect of which no international search report has been established need not be the subject of an international preliminary examination (Rule 66.1(e) PCT). The applicant is advised that the EPO policy when acting as an International Preliminary Examining Authority is normally not to carry out a preliminary examination on matter which has not been searched. This is the case irrespective of whether or not the claims are amended following receipt of the search report or during any Chapter II procedure. If the application proceeds into the regional phase before the EPO, the applicant is reminded that a search may be carried out during examination before the EPO (see EPO Guideline C-VI, 8.5), should the problems which led to the Article 17(2) declaration be overcome.

フロントページの続き

(31)優先権主張番号 00403332.0

(32)優先日 平成12年11月29日(2000.11.29)

(33)優先権主張国 欧州特許庁(EP)

(81)指定国 AP(GH,GM,KE,LS,MW,MZ,SD,SL,SZ,TZ,UG,ZM,ZW),EA(AM,AZ,BY,KG,KZ,MD,RU,TJ,TM),EP(AT, BE,CH,CY,DE,DK,ES,FI,FR,GB,GR,IE,IT,LU,MC,NL,PT,SE,TR),OA(BF,BJ,CF,CG,CI,CM,GA,GN,GQ,GW,ML,MR,NE,SN, TD,TG),AE,AG,AL,AM,AT,AU,AZ,BA,BB,BG,BR,BY,BZ,CA,CH,CN,CO,CR,CU,CZ,DE,DK,DM,DZ,EC,EE,ES,FI,GB,GD,GE, GH,GM,HR,HU,ID,IL,IN,IS,JP,KE,KG,KP,KR,KZ,LC,LK,LR,LS,LT,LU,LV,MA,MD,MG,MK,MN,MW,MX,MZ,NO,NZ,OM,PH,P L,PT,RO,RU,SD,SE,SG,SI,SK,SL,TJ,TM,TR,TT,TZ,UA,UG,UZ,VN,YU,ZA,ZM,ZW

(74)代理人 100094400

弁理士 鈴木 三義

(74)代理人 100107836

弁理士 西 和哉

(74)代理人 100108453

弁理士 村山 靖彦

(74)代理人 100110364

弁理士 実広 信哉

(72)発明者 エリー・ウジ・コスカ

フランス・F-95350・サン・ブリス・ス・フォレ・アレ・エリダン・9

Fターム(参考) 5B075 NK44 NK54 QT06