

US Patent & Trademark Application

By Jim Luisi

Date: February 22, 2001

Name: Database Management System

Abstract

A method and system for the minimization of input and output processing of a database management system is provided. A minimization method includes the automatic representation of compressed data occurrences as a one-dimensional array, redundantly represented with each bit position of the compressed data field in a distinct one-dimensional array. A one-dimensional array is a string of one or more compressed data fields, or a specific bit position of a compressed data field. In contrast to the techniques used in compressed bitmap indices, gaps are not permitted within a one-dimensional array. Depending upon the database operation required, either the entire compressed data field or only specific bit positions of a compressed data field may be required. The uncompressed representation of each data field and its compression key are stored in a normalized meta-model. The technique is applicable towards control systems and information systems software, including transaction and decision support processing involving very large databases (VLDB), with or without the aid of application specific integrated circuits (ASIC), programmable chips (FPGA) or RAID hardware technology.

Inventors: Luisi; James V. (Colts Neck, NJ)

Assignee: VPL, Inc. (New Jersey, United States)

Appl. No.: TBD

Filed: February 28, 2001

Current U.S. Class: TBD

Intern'l Class: TBD

Field of Search: 707/3, 10, 502, 2, 101, TBD

References Cited [Referenced By] U.S. Patent Documents

TBD

Foreign Patent Documents

TBD

Other References

TBD

Parent Case Text

TBD

Claims

What is claimed is:

1. A computer implemented method of representing information for read, write, update and delete operations, consisting of:
 - (a) recording the standard input and output data types and determining the compression key of each data field of each record and storing them in a meta model
 - (b) applying the compression key to each data field and representing the entire compressed data field or only specific bit positions of a compressed data field into a one-dimensional array without loss of precision in the value of the data field
 - (c) retrieval of the specific bit positions of the compressed data field as necessary to support the particular database operation
 - (d) selection criteria processing and arithmetic operations directly on the entire compressed data field or only specific bit positions of a compressed data field without decompression
 - (e) maintaining support for an unknown data value occurrence for each data field
 - (f) maintaining time series information for each data field value occurrence
 - (g) performing join processing on key and non key data fields using the entire compressed data field or only specific bit positions of a compressed data field without decompression of the data field
 - (h) performing, with the aid of application specific integrated circuits (ASIC) and or programmable chips (FPGA), arithmetic processing on compressed data fields or only specific bit positions of a compressed data field without decompression of the data field
 - (i) performing, with the aid of application specific integrated circuits (ASIC) and or programmable chips (FPGA) and or RAID, data cache, buffering, string operations
 - (j) performing, with the aid of application specific integrated circuits (ASIC) and or programmable chips (FPGA), portions of application functionality that is normally implemented in software
2. The computer implemented method of claim 1.(b) and (d) and (g) and (h), further including the additional step of skipping specific bit positions of a compressed data field involving entire one-dimensional arrays for uniform bit settings, such as the lack of 'on' bits, or because they are logical impertinent to the particular operation or sub-operation.
3. The computer implemented method of claim 1, further comprising the steps of

(a) supporting individual or bulk read, write, update and delete operations on one or more compressed data fields or only specific bit positions of compressed data fields within one-dimensional arrays

(b) mirroring individual or bulk write, update and delete operations on one or more compressed data fields or only specific bit positions of compressed data fields within one-dimensional arrays

4. The computer implemented method of claim 1, further comprising the steps of loading individual or bulk read operations on one or more compressed data fields or only specific bit positions of compressed data fields contained within one-dimensional arrays into dynamic memory

5. The computer implemented method of claim 1, further comprising the steps of extraction, decompression and export of individual or bulk read operations on one or more compressed data fields or only specific bit positions of compressed data fields contained within one-dimensional arrays

Description

FIELD OF THE INVENTION

The invention relates to databases, and more particularly, to the representation and storage of detailed data used in control systems and information systems software, including transaction and decision support processing involving highly compact data technology and very large databases.

BACKGROUND OF THE INVENTION

The way that data is organized and stored in most database products has not changed from the inception of database technology, over twenty years ago. In summary, data is stored on pages, usually 4K in length, with each page having a header and a footer record. Using this approach, the data belonging to a table is physically stored as a group of attributes that belong to an occurrence of a specific row. Depending upon the width of the tables and the number of tables that are stored within the particular address space, the number of times that a given attribute may be stored within a 4K page may vary significantly. Hence, in order to retrieve a handful of occurrences of a particular attribute, it may be necessary to retrieve a large number of 4K pages.

While some database products went in the direction of data summarization techniques, such as star schemas and cubes, some went in the direction of generating bitmap indices, such as segmented and non-segmented bitmaps, in order to determine the location of the detail data.

If we go back to early computers when computers had little, if any, memory, to a period predating the first databases, programmers used flat files containing one, few, or many data fields. A flat file is an old term, which is analogous to a single record, table or array of data, where one or more fields would be stored repeated directly after one another in storage. It is an old approach, however, it is highly compact in comparison to the file organizations and physical distribution of data found in databases. Since these early flat files did not support random access, the notion of keys, not to mention prime keys, did not exist. As a result, one of the few ways to relate data fields of one flat file to those of another, was through relative record number,

meaning that the sixth occurrence of a field in one flat file, corresponded to the sixth occurrence of a field in another flat file. In terms of modern data modeling, the data that was represented by each flat file would bear a one to one relationship, and as such, would belong to the same record or table.

Similar to flat files, the invention makes use of the fact that a one-dimensional array eliminates the overhead of storage to support the architecture of traditional file organizations and particularly traditional databases. The only information stored in a one-dimensional array is the business data. It does not contain additional records, such as a header and footer record that denote such details as the resources available on the page, what records reside on the page, where they begin and continue, and what empty space may be reserved for future expansion. However, the similarity ends there.

In the invention, the one-dimensional arrays are actively managed by the meta-model. Every data field associated with every table is recorded in the meta-model with the location of its corresponding one-dimensional array containing all of the occurrences of the entire compressed data field. It also contains the location of the one-dimensional arrays associated with the occurrences of the specific bit positions of the compressed data field, and the compression key. Therefore all input and output operations are interpreted using the meta-model.

Unlike the traditional technologies, which restrict their compression to non-key attributes, the database invention includes data fields that participate as keys in its compression, and can perform queries and arithmetic expressions without decompressing the data. The reason traditional database technologies require that key fields remain uncompressed is that they generally must decompress compressed data in order to understand and utilize the data content.

The data recorded in the database invention contains the full precision for each occurrence of data field that stored in the database. Although the full precision for each data field is recorded in the database, it is not required to support every database operation. When permitted by the particular operation or sub-operation of the database, the ability to selectively process bits of a compressed data field. Thus, certain selection criteria processing and arithmetic operations can be performed directly on the entire compressed data field or only specific bit positions of a compressed data field. Compressed data fields with specific bit positions that are uniform for all occurrences of their value for a particular data field do not require the corresponding one-dimensional array to be generated or searched in a read operation.

Update operations can maintain various levels of time series information ranging from denoting a value for the data field occurrence that is different than the previous value, to denoting the series of prior values that existed for each occurrence of the particular data field, including their effective dates.

Join processing is performed on key and non key fields using the entire compressed data field or only specific bit positions of a compressed data field without requiring decompression of the values contained within each occurrence of the data field. As join strategies are determined with the information that is maintained by the meta-model, the appropriate one-dimensional arrays are evaluated to identify the result set. The particulars relating to the various join algorithms used in conjunction with this database invention represent a distinct set of inventions.

The process of performing programmatic algorithms on compressed data fields or only specific bit positions of compressed data fields embedded within ASIC or programmable chips (FPGA) without requiring decompression is an alternate implementation approach. The algorithms

implemented in hardware are the same as in software, with the exception that algorithms implemented in software are executed by the processor that is central to the computer, as opposed to external processors.

The representation and manipulation of compressed data fields or only specific bit positions of a compressed data field as one-dimensional arrays is supportable in dynamic memory, disk drives, with or without RAID, or any other storage medium, independent of computing platform, operating system, processor type or processor speed.

SUMMARY OF THE INVENTION

A meta-model, which contains the translation rules of the data represented in the data portion of the database, provides a unique and enhanced ability to compress all types of data. By storing only compressed data fields in one-dimensional arrays, many more occurrences of data fields may be recorded within a finite area of storage, rendering a significant advantage over standard flat files.

The use of specific bit positions of a compressed data field in a one-dimensional array provides a unique and enhanced ability to minimize the input and output processing of a database for a variety of field comparisons and arithmetic operations.

Furthermore, the ability to process key and non-key data, while retaining it in its compressed form provides a unique and enhanced ability to perform a variety of comparisons and arithmetic operations without the significant expense of decompressing the values of each data field occurrence. Whether the processing is implemented by software, application specific integrated circuits (ASIC) or programmable chips (FPGA), the notion of performing these operations on data in its compressed state provides a unique and enhanced ability for processing a greater number of operations on more information in a unit of time.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

A method for representing data as compressed data occurrences within a one-dimensional array is described. In the following description, for the purposes of explanation, various details are set forth in order to provide a more thorough understanding of the invention. It will be apparent, however, to one skilled in the database industry that the invention may be practiced without these specific details. For instance, a variety of compression algorithms can be used to determine the physical representation of the detail data contained in the database, and procedural logic may be implemented using software, such as C++, or firmware, such as an FPGA board.

In the following example, we will discuss creating a derived attribute, which is defined by an arithmetic formula involving three attributes. For our purposes, we will assume that there is an attribute, attribute-A, which was defined as an unpacked integer ranging from zero to a maximum value of sixty thousand. In the meta-model, the data type of attribute-A may be defined as any of the valid numeric types. This data type identifies the external format for the attribute when entering the load process, or any process involving the exporting of data externally. The internal data type of attribute-A would be the least number of bits (LNB) necessary to represent the largest value defined for attribute-A. This compressed data field may be represented as either a contiguous string of bits, with each occurrence of the LNB immediately following the one before it. The compressed data field may also be represented as a

non-contiguous string of bits, with each bit position of the LNB in a distinct one-dimensional array.

In this manner, a non-contiguous string of bits would be stored with all occurrences of the each bit position of the LNB immediately following the one before it in distinct one-dimensional arrays. Using a numeric data type ranging from zero to a maximum value of sixty thousand, the LNB would require no more than sixteen bit positions, which may be split into sixteen separate one-dimensional arrays. Hence, if one billion occurrences of attribute-A existed, the total number of bits in a one-dimensional array containing the contiguous string of bits would be sixteen billion bits of storage, while each of the sixteen non-contiguous string of bits would result in one-dimensional arrays of one billion bits each.

Depending upon the particular selection criteria for a where clause, or the particular mathematical operation, all bit positions of an attribute may not be required. For example, if the selection criteria of a query requested the occurrences of attribute-A that were in excess of an integer value of thirty thousand, only bit positions thirteen, fourteen, fifteen and sixteen would have to be evaluated, limiting the particular operation to four billion bits of data.

NON-RELATIONAL DATABASE SYSTEMS

The present invention is not limited to any particular type of database system, product or architecture. Although the terms used herein are consistent with relational database technologies, the description of this invention could have been rendered in the terminology of any database technology. Some of the other database technologies include, but are not limited to, hierarchical, linked list, inverted list, object oriented and geographic information system (GIS) database architectures and technologies. Therefore, instead of utilizing the term database 'table', a variety of alternative terminologies could have been substituted, such as 'records', 'rows', 'tuples', 'segments' or 'object classes'.

The foregoing descriptions that illustrate the invention have been rendered with reference to specific embodiments. The techniques and representations, however, implemented, may be readily identified as being within the scope of this invention, and hence the above descriptions are intended to be regarded as one of numerous ways to represent this invention, rather than in a restrictive sense.