Disk Based Hash Tables and Quantified Numbers

Edscott Wilson Garcia

March 24, 2014

Abstract

This paper describes the design of Disk Based Hash Tables: n–dimensional self–indexed data tables. The construction of quantified numbers and their relation to the natural numbers is outlined. Application of quantified numbers to multidimensional data record keys is explained. Finally, quantum mechanics electron orbital energy levels are shown in quantified numbers with associated diagrams.

Introduction

What would be the difference between a data base application, such as MySQl, and Disk Based Hash Tables (DBH)?

Databases, such as MySQL and SQLite, organize data in tables, which are two dimensional objects with rows and columns[5]. Each row contains a unique access key and a data record. The data record need not be unique, and contains one or more fields. Records are accessed by moving up or down the table until the record index key matches that which was requested. On large tables, this is quite inefficient, so databases rely on an additional index file. The index is a smaller table with a single field which stores the offset to the actual record in the larger table. If the index is small, the entire table will reside in memory. If the index is not small enough to fit in memory, the gain achieved from using an index is relative.

DBH[8, 9] evolved from the inefficiency of conventional database tables to deal with the output of real time air pollution data in the metropolitan area of Mexico City on limited hardware[2]. In this situation, the record size was of secondary importance with regard to the number of records involved. Conventional index files could not speed up the access time to acceptable levels. To solve this problem, the topological homeomorphism[3, 4] which exists between $\mathbb{R}^2$ and $\mathbb{R}^n$ was used to solve the problem in a more efficient manner. Any two dimensional table has a n dimensional topological equivalent. This is fundamental to DBH efficiency. In essence, any two dimensional table can be optimized into a binary tree[6]. Taking this concept one step further, the same table can be structured into an *n–ary* tree. Thus, if a binary tree lies in two dimensions, an *n–ary* tree would be the n dimensional analog.

Thus, a DBH file is an n–dimensional table or *n–ary* tree. To avoid the disk head moving back in forth over the mechanical magnetic medium, the index is incorporated into the file format itself. In this fashion, the access key is split up into the desired dimensions of the table, and the search converges rapidly by moving through the pertinent dimensions.

The initial design of the multidimensional tables[1] solved several problems which arise with large tables, minimizing disk storage space without resorting to compression techniques. When a record has a large number of fields but only a subset of these contain actual data, ordinary tables tend to get crammed with useless bytes. Optimization based on compression techniques may depend on the randomness of such events. Lower level variable records proved more efficient in the development of applications targeted at the pollution problem in Mexico City[7].

The design of Disk Based Hash Tables

The following considerations were made in the design of DBH.

1. The dimensionality of each node should be variable, but less than or equal to the root node.
2. Each node shall define a n-table of dimensionality less than or equal to the root node.
3. Nodes shall not carry redundant dimensions.
4. The data record size on each node should be variable.
5. Functionality for creation, modification or deletion of arbitrary nodes should be present.
6. Repetitive commands (foreach) must be supported for the main tree or any subtree.

The design goals were acomplished with DBH by varying the dimension of the node according to the depth level. The root node has n dimensions, with a pointer in each dimensional direction. All nodes which lie in the path of the first dimension have at most one dimension. Those on the second dimension path may have up to two dimensions and so forth up the the *nth* dimension. Each subsequent node is recursively similar. With this design, inferior nodes need only the part of the access key which differs from those nodes on the superior level.

Each access key is separated into n parts, one part for each dimension. For example, if an hour key is to be divided into nine dimensions, and the maximum key is defined at ten years, the key may be expressed as a tetradecimal number. This would give 111121230 for ten years ($24 \times 365.24 \times 10 = 876609$).

Each of the digits 111121230 represent a value of the respective dimension, each symbol may only have one of four values. Thus, the maximum read operations in each dimension is 4. The worst case scenario for 876609 records would be 36 reads. This contrasts widely from the worst case scenario for a two dimensional table, where the reads could reach 876609. A speedup of $2.4E+04$.

With n–dimensional tables, the worst case scenario for read operations is given by the function $f(j) = jm^{1/j}$, which has a minimum at $j = \ln m$. Thus for 2 million records, $j \approx 14$. While for 7 billion records (current world population), $j \approx 25$. DBH is capable of handling up to 256 dimensions. The disk space in bytes which is occupied by the multidimensional formating scheme of DBH on a 64 bit system can be obtained with the following formula, where V is the maximum value on each key digit:

$$8 \sum_{n=1}^{j} \left[(j+1-n)(V^n - V^{n-1}) \right] .$$

The formation of keys by changing the base of a number from decimal to a more appropriate base works fine as long as the number of records is limited. But what happens if the number of records grows past the limit? This is where keys based upon quantified numbers are of utmost importance.

DBH record manipulation

DBH operates on a single memory block. This block has a data segment which is of variable size. The default value for the data block is 1024 bytes and may be redefined by the user with the `dbh_set_size()` call. The size of the data block should be at least the size of the largest data record. The key to the specific record the memory block refers to should be set with `dbh_set_key()` and can be retrieved with the `DBH_KEY()` macro. The data block can be set with `dbh_set_data()` and a pointer to this block may be retrieved with the `DBH_DATA()` macro. The size of the data record which will be stored on disk must be set with `dbh_set_recordsize()`. If the size of the data record exceeds the allocated space for data, `dbh_set_size()` should be automatically called.

To load, create, update or delete a specific record, this first must be located within the DBH file by means of setting the key for the DBH memory block.

In order to optimize speed, logical structure should match physical structure on disk. This is attained by regenerating the DBH file with `dbh_regen_sweep()` which will also free space allocated to erased records.

As can be seen from algorithms 1–3, creating, deleting, updating or replacing records is simple and straight forward. The power in DBH lies in the the method to locate any record in particular. This is shown in algorithm 4.

Algorithm 1 *load*

```
locate key record
if (not found) then
 return FALSE
else
 read record size
 read bytes into data
 return TRUE
fi
```

Algorithm 2 *erase/unerase*

```
locate key record
if (not found) then
 return FALSE
else
 toggle erased flag
 return TRUE
fi
```

Algorithm 3 *create/replace*

```
locate key record
if (not found) then
 get EOF pointer
 update end of dimension pointer
 write data record at EOF pointer
else
 if (new record size <= retrieved record size ) then
  write new record size
  write data record at current pointer
 else
  update invalid_record flag
  get EOF pointer
  update last dimension pointer
  write data record at EOF pointer
 fi
fi
```

Algorithm 4 *locate*

```
if (i == dimensions) then
 if (dimension_index == key_index(i) then
  return TRUE
 else
  return FALSE
 fi
fi
for dimension=i to dimensions do
 if dimension_index == key_index(i) then
  recurse(i+1)
 else
  if (no next record in dimension) then
   return FALSE
  else
   next record in dimension
  fi
 fi
done
```

Limits on our computers

The basic unit of all computational machines is the bit. This can be interpreted as the magnetic polarity of a item which can be addressed with an positive integer. This item may be a piece of silicon or even be electron. Every computer uses binary numbers. The accustomed usage of decimal based hindu numbers is arbitrary and is a legacy of our first computational machine (the fingers on our hands). Since binary notation requires many digits, hexadecimal notation is preferred, since this format allows for quick mental translation to binary format.

Combining bits we can construct any integer we desire, limited only by the number of bits we are allowed to use. Processor architectures are designed around this parameter and define the size of the arithmetic and logic registers. Beyond this size we cannot represent an integer. If we require negative as well as positive integers, the maximum value is halved, although the cardinality of the set —or number of elements— remains the same.

In order to approximate real numbers ($\mathbb{R}$), our computers will express them as a rational number in the interval $[0, 1]$ multiplied by a power of ten (scientific notation).

The first set of numbers to be conceived was the set of natural numbers ($\mathbb{N}$) for the most basic operation: counting. As such, the set is defined as $\mathbb{N} = \{1, 2, 3, \dots\}$. Zero is not included, because zero is a concept[1]. Once the operations of addition and subtraction were conceived, zero (additive identity) and negative naturals (additive inverse) were added to define a new set, the

[1]Ten apples are apples, but zero apples is not an apple.

integers, $\mathbb{Z} = \{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\}$. For multiplication and division, a new set of numbers, isomorphic to the set of natural numbers, had to be defined. This set is called the rational numbers ($\mathbb{Q}$). Rational numbers are all those which may be expressed as a quotient p/q, where p is an integer and q is a natural number.

It can be shown that there are just as many integers as there are rational numbers, and there is a one to one relationship between them. In other words, given any finite set of rational numbers, these can be written as a set of integers. Thus our computers can work reliably with a subset of rational numbers.

In the eighteenth century, Leibnitz and Newton raised the flag of the differential revolution, summing up the work of many mathematitians into calculus. The value of calculus was quickly recognized as Newton showed how to determine the motion of the planets with an accuracy until then unheard of. Since then, scientists around the world have been applying differential and integral models to just about every phenomena which varies with respect to time. This leads to differential and integral equations which for the most part, cannot be solved exactly.

In order for calculus to exist, a second type of fundamental numbers must exist, the irrational numbers. These numbers cannot be rationalized. In other words, there is no way they may be expressed as p/q, with p, $q \in \mathbb{Q}$. Irrational numbers can be viewed as a series of digits which goes on to infinity. Each digit of a single rational number can be mapped in a bijective relationship with the set of integers. In other words, the digits —or symbols— in an irrational number is countable.

π, the most known irrational number, is the ratio of the diameter to the circumference of a perfect circle —and has been approximated for milenia. Irrational numbers can only *only* exist in nature if the continuum of matter is taken for granted, *i.e.*, when nature is composed of an infinitely divisible mixture of fundamental elements. If, on the other hand, Democritus' atomic theory is accepted —*i.e.*, nature is composed of indivisible atoms and void space— irrational numbers cannot exist in nature. As an example, no perfect circle can ever be created, since in the end the circle will be nothing more than a equilateral polygon made up of atoms, and π would be a rational number which depends on the number of atoms that make up the polygon.

Even though irrational numbers exist only in the mathematical theory, they are certainly useful, just as complex numbers are. But we should never lose sight of the fact that our computational machines can only aproximate them. At some point we must truncate the infinite series of digits which define each and every irrational number. And although we can get as close as we want to any irrational number with a rational number, there are so many irrationals that they cannot even be counted. This contradicts the atomic theory, since the atoms in the universe, even if they be infinite in number, may be counted.

Quantified Numbers

Although numbering each record consecutively in decimal, hexadecimal or even binary base is a concept easily grasped and immediate to the human mind, for access on our computational machine it is not optimum, especially when the record size is not uniform. An efficient access method is by means of quantified numbers.

Say we have a series of data records. These data represent some phenomena from the universe which surrounds us. In mathematical terms the atomic theory implies that each atom in the universe can be counted and assigned a number in the set of natural numbers ($\mathbb{N}$). Since the atomic theory of the universe has grown widely accepted since the eighteenth century, we may consider that each data record can also be unequivocally addressed with an element of the natural numbers. This would start with 1.

There is a limit to the amount of natural numbers which may be exactly represented in our computers (2^{64} on 64 bit architecture). This amount is called the cardinality of the finite set of representable natural numbers. Since computers can only operate on a finite subset of the rational numbers, but p and q in the ratio p/q may have integer and natural values, the rational subset on which the computer operates is poked full of holes.

Quantified numbers, on the other hand, are defined considering the characteristics of our computational machine. In other words, they are defined considering that our machine can only work with a *finite* subset of the natural numbers.

Definition. The quantified numbers are an isomorphism of the natural numbers expressed in a different notation, as follows:

- Natural numbers of base b are expressed with n digits and b symbols, b is fixed and n may grow indefinitely.
- Quantified numbers of base b are expressed with b digits and n symbols, b is fixed and n may grow indefinitely.

The main characteristic of quantified numbers is the number of digits, or base. Just as natural numbers may be expressed with any finite base, so can quantified numbers. Although the cardinality of the symbol set in a quantified number is infinite, when applied to a finite subset of the natural numbers —which is what we have in our computers—, the cardinality of the symbol set is also finite.

Where quantified numbers differ radically from natural numbers is with respect to order. Quantified numbers of base 2 have the same order as natural numbers, but for a higher base the order may differ. In order to define order for quantified numbers the concept of *quanta* is introduced.

Definition. Quanta is the sum of the symbol representation of the number. The maximum quanta is the cardinality of the symbol set defined by the base.

Definition. The magnitude of a quantified number is obtained from its quanta.

Since several quantified numbers may have the same quanta, a second rule may be introduced to establish a unique order. This second instance rule may be —for example— a traditional right to left symbol value. Ordinary numbers are ordered with respect to this traditional right to left symbol value. This secondary rule is not necessary for data retrieval algorithms in DBH.

Summing up, for a quantified number, the cardinality of the symbol table is equal to maximum quanta in the set, while for a non–decimal base number, the cardinality of the symbol set is defined by the base of the number system. Furthermore, quantified numbers always have the same number of digits, while for natural numbers the number of digits will grow indefinitely.

Although a quantified number appears similar to a number expressed in a non–decimal base with hindu symbols, the two should not be confused.

Table 1 shows quantified numbers base 3 and natural numbers base 4 associated to the respective natural number base 10. The quantum value for each representation is also shown. The order which appears in the table is that of the natural numbers, and quantified numbers are only ordered by quanta (no secondary rule). If the secondary order criteria is applied to the quantified numbers, the order of the natural numbers is not maintained (table 2).

The cautious reader may ask whether any practical use of quantified numbers actually exists in nature. Eventhough they may be mathematically isomorphic to natural numbers, physical reality apparently is best reflected by the order of natural numbers, not quantified numbers, it so seems. In any case, quantified numbers in base 2 have the exact same order as do natural numbers in any base. But quantified numbers base 4 present an interesting application to an observed atomic phenomena which is not apparent when when natural numbers are used.

Atomic Orbitals and Quantified Numbers

Atomic theory indicates that quantified numbers clearly exist in nature. Take the quantified number set defined by base four. Table 3 shows the first 20 numbers. Table 4 shows the quanta value and the frequency with which this magnitud appears.

In the atomic theory, electrons oscilate in pairs within orbitals around the nucleus of atoms. Each energy level may contain one or more types of orbitals: s, p, d, f and theoretical g. Energy level 1 can only contain one s type orbital. Level 2 may have one s and three p orbitals. Level 3 may contain five additional d orbitals. Level 4 can fit an additional seven f orbitals. A theoretical g orbital would enter level 5, and if the series of primary numbers $\{1, 3, 5, 7, ...\}$ indicate the number of orbitals in each succesive shape, there would be eleven g orbitals to complete energy level 5. An important consideration is that energy levels are not completed sequentially, instead electrons tend to simpler orbitals in higher energy level before more complex orbitals in lower energy levels. The order in which orbitals are filled occurs as follows: 1s, 2s, 2p, 3s, 3p, 4s, 3d, 4p, 5s, 4d,

N	quantified	quanta	base 4	quanta
1	000	0	001	1
2	001	1	002	2
3	010	1	010	1
4	100	1	011	2
5	002	2	012	3
6	011	2	020	2
7	020	2	021	3
8	101	2	022	4
9	110	2	100	1
10	200	2	101	2
11	003	3	102	3
12	012	3	110	2
13	021	3	111	3
14	030	3	112	4
15	102	3	120	3
16	111	3	121	4
17	120	3	122	5
18	201	3	200	2
19	210	3	201	3
20	300	3	202	4
21	004	4	210	3
22	013	4	211	4
23	022	4	212	5
24	031	4	220	4
25	040	4	221	5
26	103	4	222	6
27	112	4	1000	1

Table 1: Values of quantified numbers base 3 compared with natural numbers base 4

N	quantified	quanta
1	000	0
2	001	1
3	010	1
4	100	1
6	011	2
8	101	2
9	110	2
5	002	2
7	020	2
10	200	2

Table 2: Values of quantified numbers base 3 ordered by quanta

Natural number	Quantified expression
1	0000
2	0001
3	0010
4	0100
5	1000
6	0002
7	0011
8	0020
9	0101
10	0110
11	0200
12	1001
13	1010
14	1100
15	2000
16	0003
17	0012
18	0021
19	0030
20	0102

Table 3: First twenty quantified numbers, base 4

quanta	frequency	energy level	orbitals	Atomic number / symbol
1	1	1	1	2 / He
2	4	2	4	10 / Ne
3	10	3	10	30 / Zn
4	20	4	20	70 / Yb
5	35	5	35	> 102 / No

Table 4:

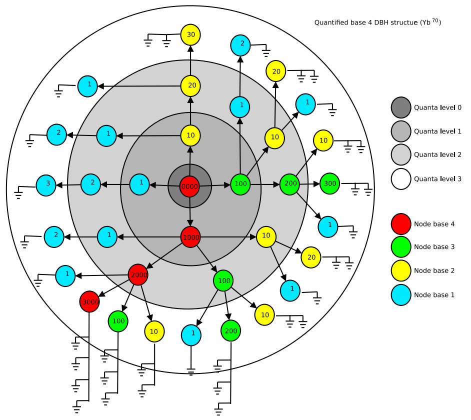

Figure 1: Atomic Yb electronic structure and associated DBH structure

5p, 6s, 4f, 5d, 6p, 7s, 5f, 6d, 7p. Table 4 shows the orbitals that must be filled with electrons in order to complete each energy level, alongside the frequency for each respective quanta in base 4 quantified numbers. The match between the number of orbitals and quanta frequency is exact.

This takes us back to the original Niels Bohr atomic model, where electrons would fill energy levels, and these energy levels are ordered as the natural numbers (*i.e.*, the order for base 2 quantified numbers). The only precision to make is that the order that should be followed is that defined by the base 4 quantified numbers.

Base 4 quantified keyed DBH structures for the atomic elements Yb and U are shown in figures .1 and 2. This demonstrates that quantified numbers actually exist in nature (at the atomic level at least) and the realm of application extends beyond the formation of DBH keys which ensures the scalable growth of balanced n–dimensional data trees.

As to the appearance of quantified numbers in nature, modern atomic theory has established that electrons are found in orbitals. Electron orbitals do not scale in energy levels as planetary orbits do. Instead, several orbitals may have the same energy level. Quantified numbers provide a simple way to associate these orbitals to their respective energy level.

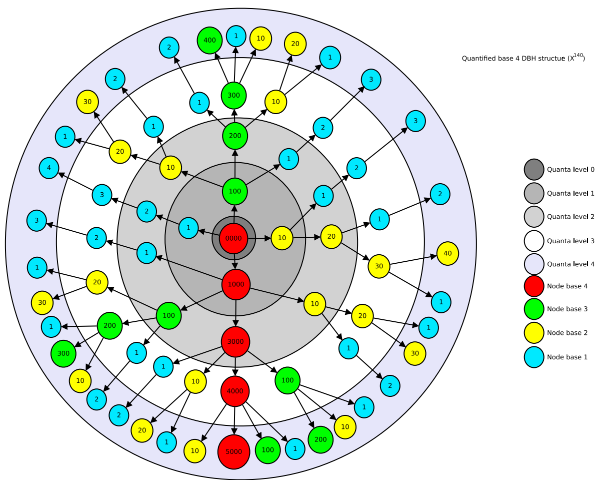

Figure 2: Atomic U electronic structure and associated DBH structure

Figure 3:

Quantified numbers applied to data retrieval

For the past decades, one of the bottlenecks which holds back the time a computational process takes to complete are the clock cycles taken to get the data from memory storage to CPU registers to perform the necessary logical or arithmetical operation plus the cycles taken to put the result back to memory storage.

A second bottleneck in modern computing, not as critical as the above, is the time elapsed to get processor instructions from memory into the CPU.

The amount of data we are capable of obtaining from any number of phenomena has always been greater than the amount of memory available in our computers. Thus we have relied on slower memory devices with much larger storage areas. In the beginning these devices were magnetic tapes, which had a number of disadvantages too great to mention here. Currently, rotating disk drives have become quite reliable and can be configured for parallel access to provide very large storage areas. With such devices, current comodity off the shelf devices are capable of handling datasets of up to 18 exabytes. Eventhough that may seem an unattainable amount of data, current oil exploration techniques can quickly fill up those buckets. Future technology, with 128 bit data addressing, will put data sets sizes at the yottabyte (2^{80}) and beyond.

So all that data can reside on storage devices. How can we minimize the access time for one particular data record? How can we define a strategy that will scale in performance as our data set size continues to grow? Solving that problem is the heart of DBH usage of quantified numbers.

Algorithm 5 *genkey*

```
input natural number (n), key pointer (key), dimensions (d)
// get quantum value for n and d
q = quantum(n,d)
// get natural number (m) for d-1 dimensions
m = natural(n, q, d-1)
//assign key dimension index
*key = n - m;
// return condition
if d==1 then
 return
fi
recurse(m, key+1, d-1)
```

DBH generation of quantified numbers

DBH has several functions to convert from natural numbers to quantified numbers. The only difference between these functions are the symbols contained in the symbol table. Function `genkey()` starts with '0', while `genkey2` starts with 'A', and `genkey0` starts with 0. Basically, `genkey2()` and `genkey()` will use `genkey0()` and offset each byte by 65 and 48 respectively. The algorithm for `genkey0()` is shown in algorithm 5. This is an efficient algorithm, but it must be noted more than one set may satisfy the conditions of quantified numbers. The choosen mapping will determine whether or not the order of natural numbers is preserved or how it is modified.

As an additional function, DBH library provides the `orderkey()` function, which maps a subset of the natural numbers to fixed digit variable base hindu number. This mapping preserves the order of the natural numbers, but for that to occur, a maximum symbol value (or base) must be specified. If you know how to preserve the order of natural numbers without specifying a base, please share.

If data records are to be accessed randomly, the order of the natural numbers is irrelevant. But as shown above, the quanta is of utmost importance to minimize the access to any record in general, not the order of natural numbers.

Comparison with GDBM

Under a first impression, DBH and GDBM[10] seem similar in functionality, but they differ in the fundamental design and the problem set out to solve. The problem DBH set forth to solve was to minimize access time to any particular record on very large data sets. The meaning of "very large datasets" is relative to the available shared memory of a system, which is almost always much less than the available disk memory. This means that the design of DBH is meant to maintain its effectiveness as newer and better computers are available.

Thus the domain of application of DBH is much wider than that of GDBM.

DBH covers the domain of GDBM, but GDBM cannot operate on the full domain of DBH. More on this below.

Similarities

- Both use a unique key to access a particular data element
- Both have variable size data records
- Both have functions to search, retrieve, modify, erase records
- Both have sequential iteration over all key/data pairs
- Both are GPL

Differences

- GDBM works with a memory hashtable to a flat data table file, with records cached in virtual memory.
- DBH is a structured multidimensional table file which takes advantage of the built in cache mechanisms built into disk devices and the OS file cache.
- GDBM is limited to $<$ 2GB records, DBH does not have this limit.
- GDBM reads everything into its cache, which efectively limits the size of the data table to the available virtual memory before swapping begins and performance deteriorates, the table size for DBH does not have this limitation.
- DBH has threadsafe mode (for concurrent operation between threads) and parallel safe mode (for concurrent operation between independent processes).
- GDBM has several cool things, which can easily be incorporated into DBH, such as the definition of a user defined fatal function, and import/export functionality.
- GDBM is two dimensional. There is only one way to go through the table, from first to end. DBH is multidimensional, you can go through the table by dimensions (sweep) or across dimensions (fanout), and each data record is a complete table in itself, so you can do selective foreach procedures.

Performance comparison

Conclusions

DBH software —in use in over 80 countries since 2002— has shown that n–dimensional data tables are an efficient way to organize large amounts of data in a scalable manner which surpases current storage device capacities and which continues to support ever larger data sets with the use of quantified numbers as a means for the creation of index keys.

Quantified numbers are a tool to work within the limited symbol representation available to our computational machines and are also a means to understand certain atomic level phenomena in a simple way.

More research on the nature of quantified numbers is necessary to grasp the full potential of this set of numbers when applied to computational science and numerical methods, as well as other branches of modern science.

References

[1] Edscott Wilson Garcia, “Diseño del sistema objeto multinodal orientado a disco para bases de datos”, *Revista del Instituto Mexicano de Petróleo*, vol. 24, num. 4, 1992. (document)

[2] Edscott Wilson Garcia and Jose Moran Gonzalez and Emmanuel Gonzalez Ortiz, “Desarrollo de un sistema para evaluar el impacto de la calidad del aire en la ZMCM”, *Revista del Instituto Mexicano del Petróleo*, vol. 26, num. 2, 1994. (document)

[3] Michael Henle, *A Combinatorial Introduction to Topology*, Dover Publications, New York, 1979. (document)

[4] George McCarty, Topology an Introduction with Application to Topological Groups, Dover Publications, New York, 1967. (document)

[5] Jay Greenspan and Brad Bulger, *MySQL/PHP Database Applications*, John Wiley & Sons, Inc. New York, 2001. (document)

[6] Satoshi Watanabe and Takao Miura, “Reordering B-tree files”, *Proceedings of the 2002 ACM symposium on Applied Computing*, pp. 681–686. (document)

[7] Edscott Wilson Garcia and Eduardo Buenrostro Gonzalez and Emmanuel Gonzalez Ortiz and Manuel Moran Gonzalez, “XDF System for Graphical Monitoring of Weather and Pollution”, *Sixth International Conference Environsoft*, eds. P. Zannetti and C. A. Brebbia, Computational Mechanics Publications, Southampton, U. K., 1996. (document)

[8] *http://dbh.sf.net* (document)

[9] *http://sf.net/projects/dbh* (document)

[10] *http://www.gnu.org/software/gdbm*

(document)