Design and Implementation of a Cache-based Granular Computing System

Xiaoxia Huang[1] Lun Cheng[2]

[1]Department of Computer, Shanghai Maritime University, Shanghai 200135, P. R. China

[2]Department of Control Science and Engineering, Tongji University, Shanghai 200092, P. R. China

Abstract

This paper presents a multilayer cache-based attribute granular computing system. It can be used to implement algorithms based on attribute granular computing. After defining an attribute granule, it brings forward an operation and storage model. Six basic mapping operations and three suitable storage patterns are designed for all kinds of attribute granular computing and storage. A cache-based storage management with schedule algorithm can improve the efficiency of data access and solve the problem which is caused by the limited memory capacity. Finally a hierarchy cache-based attribute granular computing system is implemented.

Keywords: Granular computing, Cache, Data mining, Attribute granule, Schedule algorithm

1. Introduction

Granular Computing (GrC) is a new conceptual and computing paradigm for information processing. Informally, any computing theory/ technology that involve elements and granules (generalized subsets) may be called granular computing (GrC) which was named by T.Y.Lin in 1996 [1]. Now it may be regarded as a label of theories, methodologies, techniques, and tools that make use of granules in the process of problem solving [2]. Various kinds of technologies such as fuzzy sets, rough set, shadowed sets, probabilistic sets, etc, have different contributions on granular computing [2] -[10].

Design and implementation of a cache-based attribute granular computing system is presented in this paper. The basic storage and computing elements in the system are attribute granules which are defined in this paper first. Computing between attribute granules can be viewed as one or more mapping operations we summarized. Three corresponding storage patterns: single-column pattern, multi-column pattern and mixed-column pattern are suitable for the elements too. The system with two-level cache structure assures the high efficiency of the granules exchange and great capacity of the granules storage. The system can be used in the area such as data mining application as long as the algorithms are designed based on attribute granular computing model.

2. Theory of attribute granule

2.1. Definition of an attribute granule

A granule may be interpreted as a local or a special observation of a larger unit formed by numerous small particles. A new concept we name it attribute granule is presented in this paper.

Assume that information about objects in a finite universe is given by an information table [11] below, in which objects are described by their values on a finite set of attributes.

$$M = (U, At, L, \{V_a \mid a \in At\}, \{I_a \mid a \in At\}) \quad (1)$$

Where U is a finite nonempty set of objects, At is a finite nonempty set of attributes, L is the describing language of objects, V_a is a nonempty set of values for $a \in At$, $I_a : U \to V_a$ is an information function that maps an object of U to exactly one value in V_a. Table 1 is an example of information table, taken from an example from Quinlan [12].

Definition 1: An attribute granule

An attribute granule can be defined as a projection column (or columns) according to a single attribute (or several attributes) of an information table M

$$A = \pi_a(M) \quad (2)$$

Where

A is an attribute granule, M refers an information table, π is a projection operators , a might be a single attribute or several attributes, π_a means attribute projection according to attribute a （$a \in At$）, an attribute granule $A = \{x_1, \cdots, x_m\}$ is a set of values of attribute a of the objects.

Object	Height	Hair	Eyes	Class
O1	short	blond	blue	+
O2	short	blond	brown	−
O3	tall	red	blue	+
O4	tall	dark	blue	−
O5	tall	dark	blue	−
O6	tall	blond	blue	+
O7	tall	dark	brown	−
O8	short	blond	brown	−

Table 1: An information table.

Height
short
short
tall
tall
tall
tall
tall
short

Hair	Eyes
blond	blue
blond	brown
red	blue
dark	blue
dark	blue
blond	blue
dark	brown
blond	brown

1a 1b

Fig.1: An example of attribute granule.

Fig.1 is an example of two attribute granules of Table 1. Fig.1a is an attribute granule with one attribute, while Fig.1b is a granule with two attributes.

2.2. Operations of attribute granules

Definition 2: Attribute granular computing

Computing between attribute granules is defined as

$$A_{new\,j} = OP(A_1, \cdots, A_n) \qquad j = 1, \cdots, m \quad (3)$$

Where

$A_1, \cdots, A_n$ are n original attribute granules,

$A_{new\,j}(j = 1, \cdots, m)$ refers to one of the m new attribute granules which is combined or divided by objective or subjective method.

Computing between attribute granules can be viewed as one or more mapping operations of the granular elements. Every mapping operation converts the old granule (or several granules) to one new granule (or several granules) with same (or different) scale. Six kinds of basic mapping operations are summarized as follow (showed in Table 2), their mapping processes are shown in Fig. 2.

ID	Name	Specification
1	Msse	Create a new single-column granule with equal length from a single-column one
2	Msme	Create a new multi-column granule with equal length from a single-column one
3	Mmse	Create a new single-column granule with equal length from a multi-column one
4	Mmme	Create a new multi-column granule with equal length from a multi-column one
5	Mssn	Create several new single-column granules with unequal length from a single-column one
6	Mmmn	Create several new multi-column granules with unequal length from a multi-column one

Table 2: Six basic mapping operations.

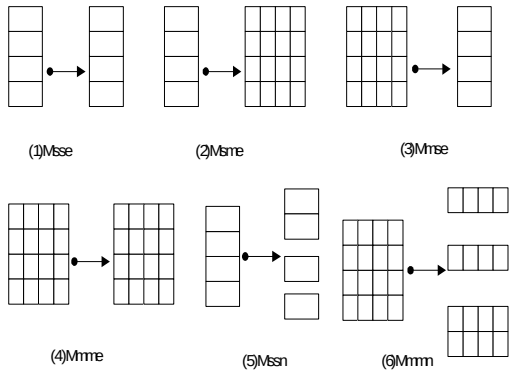

Fig.2: Mapping operations of attribute granules.

2.3. Storage patterns

Three storage patterns suitable for the single-column and multi-column attribute granules storage and

operation are presented. They are single-column, multi-column and mixed-column pattern showed in Fig. 3.

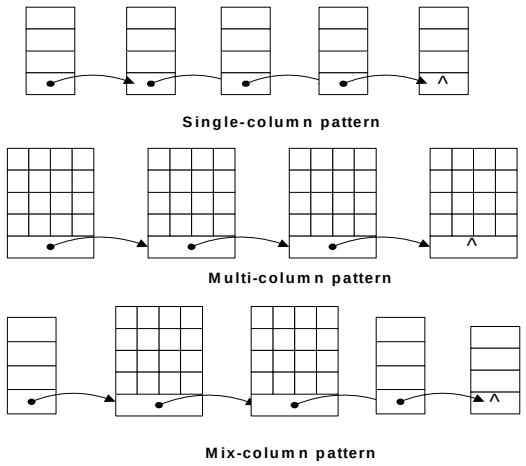

Fig. 3: Three storage patterns.

Single-column pattern is used for single-column attribute granules storage. Several single-column attribute granules can be stored in a container at the same time. Each granule is assigned a distinct ID automatically when it is adding. Multi-column pattern is used for multi-column attribute granules storage. The width of it can be defined by user. Mixed-column pattern combines the features of the above two patterns, can be used for several single-column and multi-column granules. A container of every kind of pattern has several basic elements. All data of an element store in a continuous space.

2.4. Cache-based schedule algorithm

The cache-based storage management can improve the efficiency of data access and solve the problem which is caused by the limited memory capacity. It has two cache levels which are main memory cache (as first level) and local file cache (as second level) in this system. A schedule algorithm we designed can schedule the granules between these two cache levels and database storage level.

The algorithm includes two functions: CacheManage and SwapData. CacheManage tests the signal type and call SwapData to finish job. The algorithm is shown as follow: where line 5 creates a data accessing object, line 8 creates a cache page index object and add it into a collection. Line 15 inserts data of dataInfo object into the database, line 20 save data into database. SwapData loads the data page into the main memory, then returns the query data offset in the buffer.

```
Algorithm: Cache-based schedule algorithm
// manage data among three levels storage, according to the
   signal type.
Input: datainfo  //contains the data buffer and relational
   information
        sig     // the event type
Output: parameter // return the query data offset in the buffer
//1, 0 means operation succeed, -1 means operation failed.

CacheManager(dataInfo, sig)
1     pageIndex <- 0
2     switch sig
3      case :   sig_init
4              startIndex <- 0
5              Accessor <- createDatabaseAccessor(dataInfo)
6              while 0 <> (sz <- getPage(
                 Accessor, dataInfo.sqlString, dataInfo.data))
7                  pageIndex ++
8                SaveToFile(dataInfo)
9                  addToCachePageCollection(new
   CachePage(pageIndex, startIndex, sz, getFileName))
10                 startIndex <- startIndex + sz
11             ReleaseDatabaseAccessor(Accessor)
12             return 0
13     case:   sig_write_new
14             Accessor <- createDatabaseAccessor(dataInfo)
15             executor_insert_batch(Accessor)
16             ReleaseDatabaseAccessor(Accessor)
17             return 0
18     case:   sig_write_update
19             Accessor <- createDatabaseAccessor(dataInfo)
20             executor_update_batch(Accessor)
21             ReleaseDatabaseAccessor(Accessor)
22             return 0
23     case: sig_next
24             return SwapData(dataInfo.dataIndex + 1)
25     case:          sig_pre
26             return SwapData(dataInfo.dataIndex + 1)
27     case:          sig_random
28             return SwapData(dataInfo.dataIndex + 1)
29     default:
30             return -1
31    end switch

SwapData(dataIndex)
1       cachePage = cpc.getActivePage()
2      if IsActivePage(cachePage, dataIndex) = false then
3              cpc.setActivePage(dataIndex)
4       end if
5       return cpc.getActivePage().getCursor()
```

3. Design and implementation of the System

3.1. System structure

We design a multilayer cache-based attribute granular computing system with high efficiency storage manage

subsystem. It consists of four layers shown in Fig. 4: user interface layer, the algorithm based on attribute granule layer, the attribute granular model layer and the cached-based storage management layer.

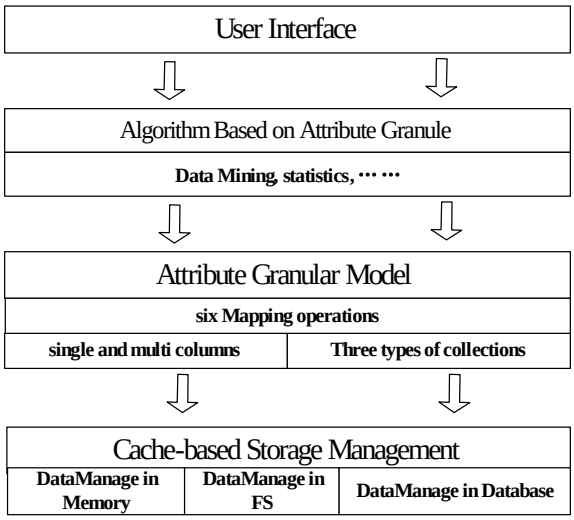

Fig.4: System structure.

The cache-based storage management layer implements attribute granules storage management based on cache schedule algorithm. It can schedule the granules among the memory, file system and database according to the input parameter and the data buffer information. The attribute granular model layer implements three attribute granular classes and six common transform operations. All the attribute granular classes supply the unify data access function (next, pre, random access). Each operation can transform one or more granular object to one or more other granular object. The algorithm layer implements algorithms based on attribute granule. The algorithms include data mining algorithms, statistics algorithms, data transform algorithms, etc. which using the attribute granular classes and mapping operations. The basic storage and computing elements in the system are attribute granules. Two-level cache structure assures the high efficiency of the granules exchange and great capacity of the granules storage. The system can be used to implement algorithms based on attribute granular computing.

3.2. Cache-based storage management

This layer consists of cache scheduler, memory management module, file management module and database management module. The input parameters of this layer are the cache pages, their status information and signal type. The cache scheduler implements management of the granules according to the input parameters by calling functions of these three management modules. The detail information is shown as follow in Fig.5.

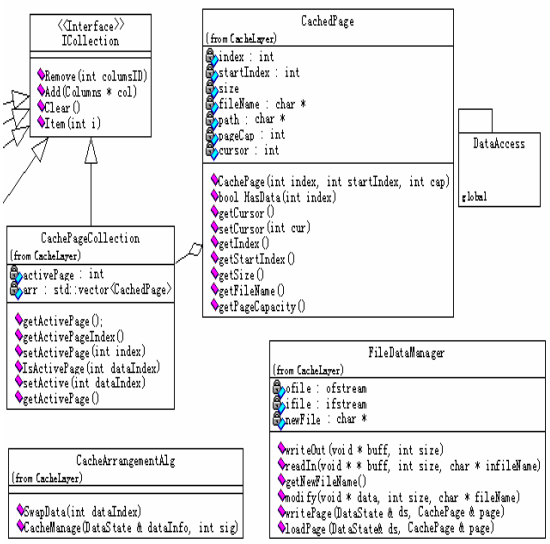

Fig. 5: UML diagram of cache-based storage manage layer.

3.3. Implementation of attribute granular model

This layer calls functions supplied by cache-based storage manage layer to implement three attribute granular classes and six mapping operations. The basic elements are single-column and multi-column granules. Three attribute granular classes supple the same interface of creation, free and some other data access functions. Three attribute granular object collections implement the basic collection algorithms, such as add, remove, clear and index. The detail of implementation is shown in Fig.6

Six mapping operations and their function interface are as follow:

(1) **Msse:**

int single_map_single_equ(single_column * single, convert_func * func, single_column ** new_single);

Suitable computing example: NOT operation of an attribute granule with Boolean type elements;

(2) **Msme**:

int single_map_multi_equ(single_column * singlecol, convert_func * func, multi_columns ** new_multicol);

Suitable computing example: divide a single-column granule by several attribute value;

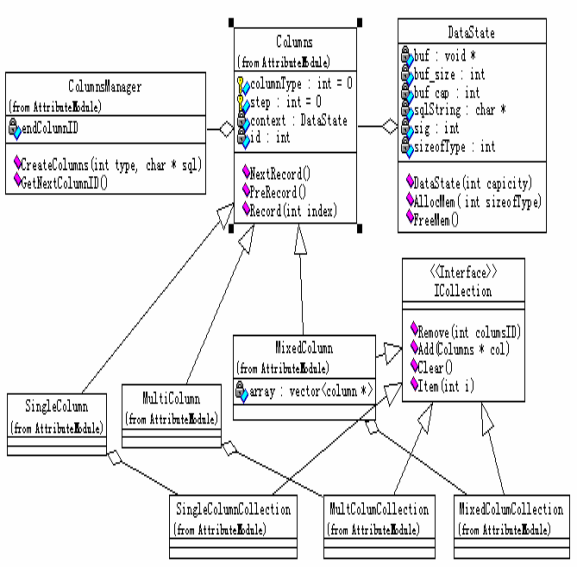

Fig.6: UML diagram of attribute granular classes and its collection.

(3) **Mmse:**
Int multi_map_single_equ (multi_columns * multicol, convert_func * func, single_column ** new_singlecol);
Suitable computing example: Intersect and Union operation of attribute granules with Boolean type elements;

(4) **Mmme:**
int multi_map_multi_equ(multi_columns * multicol, convert_func * func, multi_columns ** new_multicol);
Suitable computing example: Unitary operation for multi-column attribute granule;

(5) **Mssn:**
int single_map_single_nequ(single_column * singlecol, convert_func* func, int ncount, single_column_collection ** new_singlecols);
Suitable computing example: Summary statistics for single-column attribute granule;

(6) **Mmmn:**
int multi_map_multi_nequ(multi_columns *multicols, convert_func* func, int ncount, multi_columns_collection ** new_multicols);
Suitable computing example: Summary statistics for multi-column attribute granule.

In the interfaces of these six basic operations above: the "convert_func *func" is a function pointer, the declare of the pointer is: typedef int (*convert_func) (void * granule_from, int index_from, void ** granule2); the function is used to convert element(s) of a granule into new element(s) of an another granule (or granules), according to the mapping operation interface which is selected by the customer.

3.4. Algorithm based on attribute granule

The attribute granular computing can be used in many areas, such as data mining, statistics, data transform and so on. Algorithms need to be designed according to the characteristic of basic attribute granules and operations. For example, the k-means algorithm of the data mining can be described as below: First, there is a multi-column attribute granule A_m; with Mmse operation on A_m a mid-result As which means distance between each record and the center record is created ; then adjust the center according to requirement and continue use Mmse operation again and again until requirement conformed; using Mmmn operation to get the final result with dividing A_m into many multi-column granules according to the mid-result A_s. Algorithms designed with attribute granules have high computing efficiency, we will discuss in the future.

4. Conclusions

A multilayer cache-based attribute granular computing system with high efficiency storage manage subsystem is designed and implemented in this paper. A definition of an attribute granule is presented first. Six basic mapping operations are designed for attribute granular computing. Three storage patterns: single-column pattern, multi-column pattern and mixed-column pattern can storage basic granular elements. Cache-based schedule algorithm is given. Finally the system structure and the implementation of every level are given. The system can be used to implement algorithms based on attribute granular computing. Further research will focus on modeling the data mining algorithms based on attribute granular computing.

Acknowledgment

This work is supported by Technology Project (Grant No.05FZ22) and Technology Pivot Project (Grant No.06ZZ42) by Shanghai Municipal Education Commission.

References

[1] T.Y. Lin, Granular Computing II: Infrastructure for AI-Engineering Examples, Intuitions and Modeling. *Proceeding of the 2006 IEEE International Conference on Granular Computing, Atlanta, USA,* pp.2-7, 2006.

[2] Y.Y. Yao, Granular computing: basic issues and possible solutions, *Proceedings of the 5th Joint Conference on Information Sciences, Vol. 1. Atlantic City, NJ, USA: Association for Intelligent Machinery*, pp.186~189, 2000.

[3] Y.Y. Yao, Granular computing for data mining, Proceedings of SPIE Conference on Data Mining, *Intrusion Detection, Information Assurance, and Data Networks Security*, pp.1-12, 2006.

[4] Y.Y. Yao, Wong SKM, Wang LS, A nonnumeric approach to uncertain reasoning. International, *Journal of General Systems*, 23(2):343-359, 1995.

[5] L. Polkowski and Skowron. A., Towards adaptive calculus of granules, *Proceedings of 1998 IEEE International Conference on Fuzzy Systems*, pp.111-116, 1998.

[6] T.Y. Lin, Granular computing, announcement of the BISC Special Interest Group on Granular Computing, 1997.

[7] L.A. Zadeh, Towards a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic, *Fuzzy Sets and Systems*, 19:111-127, 1997.

[8] Y.Y. Yao, Ning Z., Granular computing using information table, In: Lin TY, Yao YY, Zadeh LA, eds. Data Miming, *Rough Sets and Granular Computing*, pp.102-124, 2000.

[9] L.A. Zadeh, Fuzzy logic=computing with words, *IEEE Transactions on Fuzzy Systems*, 4(1):103-111, 1996.

[10] LA. Zadeh, Some reflections on soft computing, granular computing and their roles in the conception, design and utilization of information/intelligent systems. *Soft Computing*, 2(1):23-25, 1998.

[11] Z. Pawlak, *Rough Sets, Theoretical Aspects of Reasoning about Data*, Kluwer Academic Publishers, Dordrecht, 1991.

[12] J.R. Quinlan, Learning efficient classification procedures and their application to chess endgames, *Machine Learning: an Artificial Intelligence Approach*, pp. 463-482, 1983.