THE PILE SYSTEM

Milan Tomeš
Univerzita Pardubice, Fakulta ekonomicko-správní, Ústav systémového inženýrství a informatiky

Abstract: *The Pile system brings new alternative, progressive and especial look at data. It works with relations instead of data, and that is why storing it without redundancy. This paper describes basic features, principles and possible applications of this system.*

Key words: *Pile, relation, tree*

Introduction

In current computing, data is stored redundantly, i.e. each recurring string, be it within a document or in different documents in the same or in different files/folders is stored again. This is not only an expensive method in terms of required storage capacity, but also in search time, since a stored string has no implicit connection to its recurrent representations, so every document and every file/folder has to be searched and compared with a given query string (unless some kind of an index is additionally provided).[1]

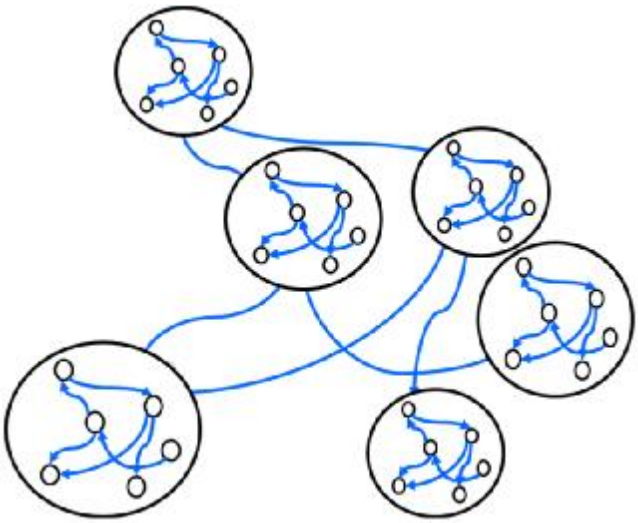
To exemplify this, we take a sentence like “The rain was over and the sun was shining again” redundancy is significantly. The letters „a“ and „n“ occurs 6x, „s“ occurs 4x, pair „in“ 4x etc.

With entire paragraphs, documents or data collections, the number of recurring substrings (like words, letters) explodes, since the number of characters used in ASCII text is just 256, and the number of words used in a language are thousands.

By contrast our nervous system works with relations. Each interaction of an organism with its environment is represented in the nervous system as a relation of some states of activity. Treating these relations as independent entities and interacting with them (creating new relations with old relations) constitutes *thinking* [4].

PILE also works with relations: a PILE connection (see Page 202) is a relation between two other PILE connections, it is a referable object and can be treated as an independent entity.

So we can present PILE as Networks of relations (Picture 1).



Picture 1: Networks of networks of relations[2]

A PILE system possesses some features, which are a necessary condition for creating systems analogous to cognitive systems with the following characteristics [1]:

a) Connectivity – any two objects in a PILE system can be connected in PILE

b) Heterogeneity – different types of terminal values (see page 203) can be connected in a PILE structure

c) Self-reference – the dynamics of the system is specified by the system itself. PILE can be used as data architecture for modeling self-referential systems.

d) Distributed structure – while having many roots and system definers (see page 203), a general PILE structure is a distributed structure. It can also be described as uniform: In a neighborhood of any object it looks the same, each object has two parents and eventually a number of children.

e) Growth/scalability – a PILE system can grow extending its structure by adding new PILE connections. A part of a PILE system would have the same growth dynamics (rules) as the whole system would.

The features that distinguish PILE from other data structures are [1]:

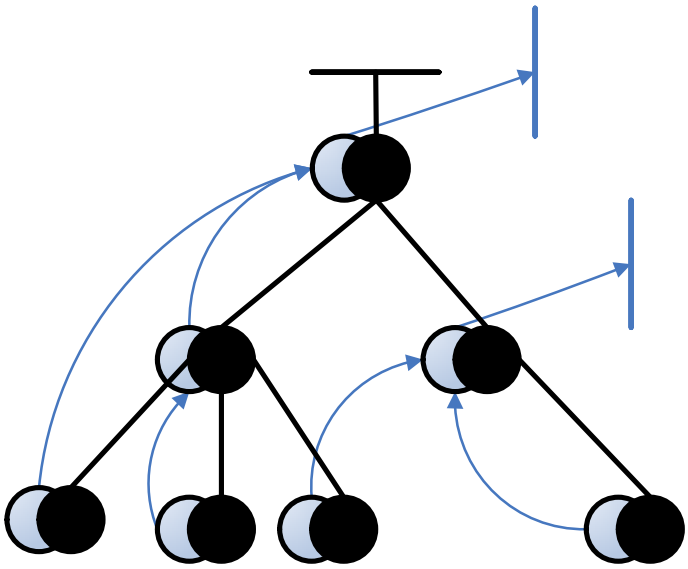
a) Multiple inheritance – each PILE object has two parents (see page 202).

b) Connections and objects are the same instances – the principle of recursivity is used up to its limits.

c) Many roots – there can be many “beginnings” of the system and many internal hierarchies.

Definition:

A PILE structure is a graph, which can be described as a combination of trees. A PILE object is an identification of two nodes from different trees, see Picture 2.



Picture 2: Pile objects

There is the two trees, black line we call **normative system**, blue arrow we call **associative system**. Each node from the normative system is identified with a node of a tree from the associative system. [1]

Drawing the associative system we put an arrow to each connection, because there is no more reflection of the hierarchy in the picture (roots on the top, leaves at the bottom). Note that the arrows point from the child to the parent. This is a PILE convention. [1]

Definition 1

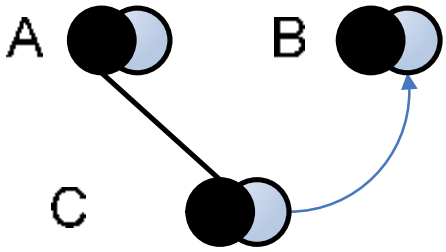
Let there be two sets of disjoint trees. We call them **normative** and **associative** generating systems, respectively. [1]

1. A **PILE structure** is a combination of the normative and the associative generating systems, so that each normative node is identified with exactly one associative node and *vice versa*.
2. A pair of nodes, a normative and an associative, which are identified in the PILE structure, is called a **PILE object**.
3. Only such structures are allowed, by which every (ordered) pair of objects has not more than one common child.

Note: The condition 1 ensures that each PILE object has exactly two parents (an ordered pair of objects). The condition 3 says that a child is uniquely defined by its parents. [1]

Definition 2

We say that an object A is directly connected to an object B in PILE, if there is an object C, which is a normative child of A and an associative child of B (Picture 3) [1].



Picture 3: Connecting objects

By the definition of the PILE system (condition 3) this object C is unique. We say, the object C is the PILE connection between the objects A and B.

Note that there can be another PILE connection between the objects A and B, where B is the normative parent and A is the associative parent.

Every PILE object, if it is not a root, connects exactly two other objects in PILE (its parents). So the PILE objects and the PILE connections are the same instances.

If we have a PILE structure in which two objects are not directly connected in PILE, we can always extend the PILE structure by a new object/connection, so that they are connected: we add a new object, which is a normative child of the first object and an associative child of the second object. The objects are connected with respect to their order, so there are always two ways to connect them: ether the first object is the normative parent and the second is the associative parent, or vice versa.

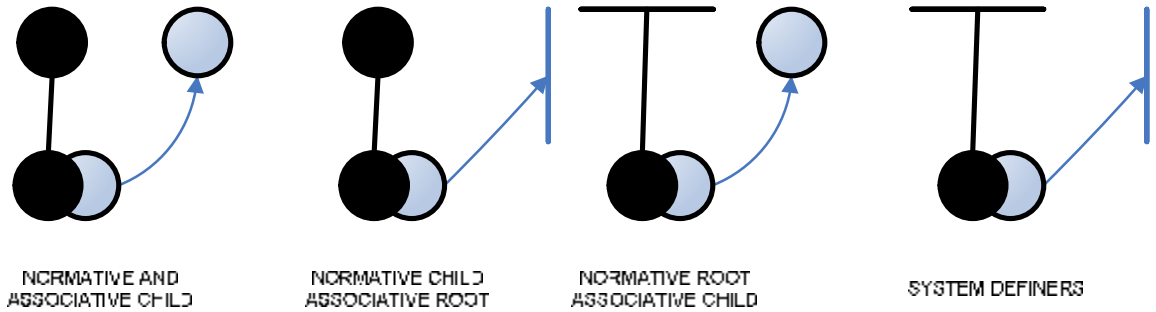
In general, any object in a PILE structure can be directly connected to any other object. [1]

Roots and terminal values

Terminal values are the interface between a PILE structure and the outside world. They can represent any kind of data entities (for example an alphabet) or any mixture of data types.

Each PILE object, which is not a root of a tree, is defined by its two parents – by the definition of the PILE structure there are no other objects having the same parents. Roots have terminal values as their “parents”. We say that a root represents a terminal value.

If, for example, a PILE object is an identification of a normative child and an associative root, then it is defined by its normative parent and it represents one special terminal value. Objects, which are normative and associative root at the same time, represent two terminal values. They are called **system definers** [1]. All this objects show Picture 4.



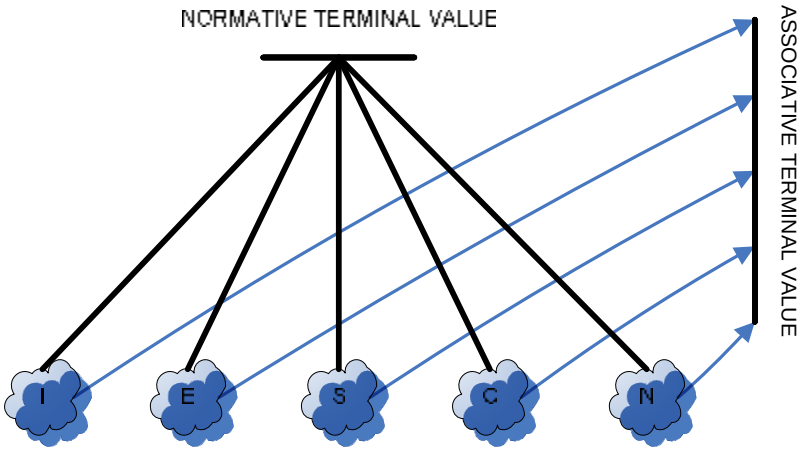
Picture 4: Types of objects

In Picture 2 there is just one normative tree, and its root is at the same time a root of an associative tree. So we deal here with a system definer. And there are two associative trees: one of them has the mentioned system definer as a root, the root of the other belongs to an object, which is a normative child of the system definer. [1]

Example:

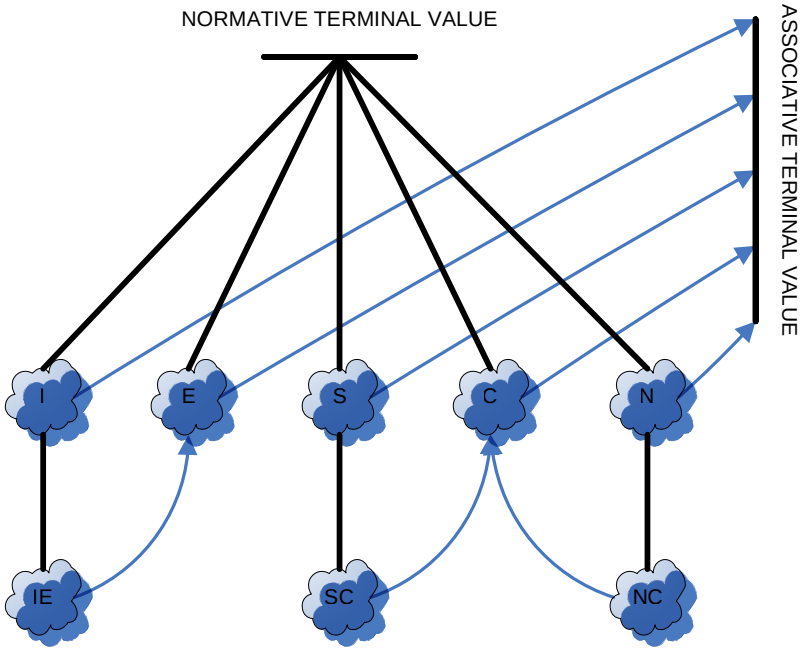
Here is an example, that build Pile system for string ”SCIENCE”. This is only one of many possibilities of representation this string.

At first we have to create system definer. In this case we have one normative tree and five associative roots, one for each symbol (“C”, “I”, “E”, “N”, “S”), see Picture 5.



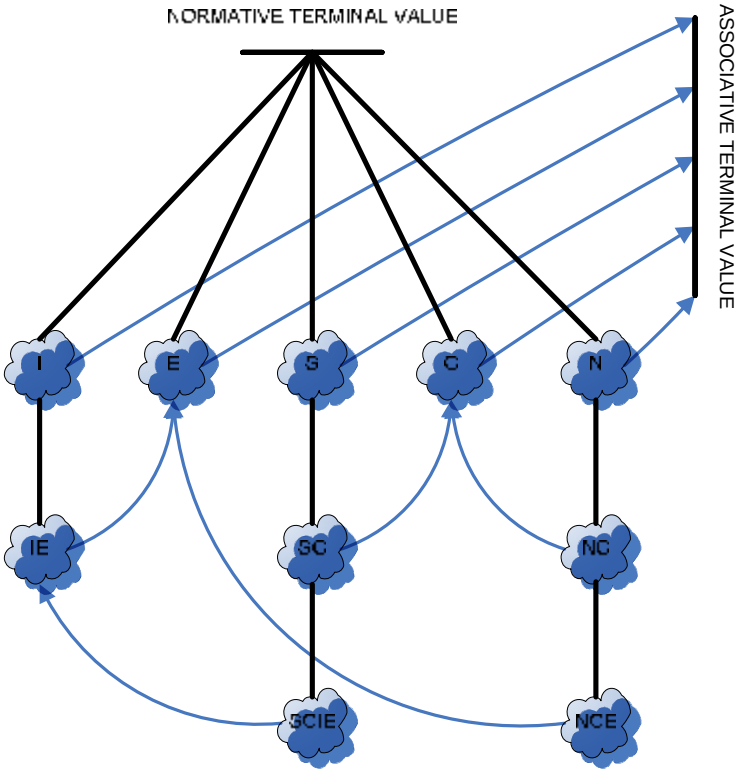
Picture 5: Example – step 1

We connect the roots representing “I”, “E” via a new object, which will be represents string “IE”. This object is a normative child of “I” and associative child of “E”. In the same way we connect objects representing “S” with “C” and “N” with “C”. This shows Picture 6.



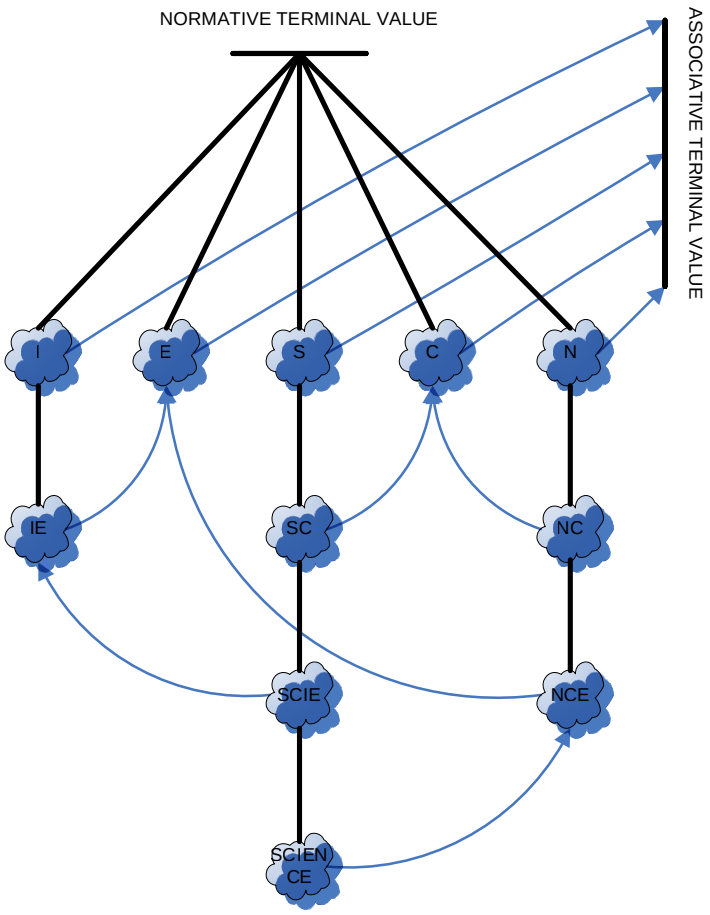
Picture 6: Example – step 2

Now we connect new objects “SC” with objects “IE”. By this we get objects “SCIE”, that is normative child of objects “SC” and associative child of objects “IE”. In the same manner, we get objects representing “NCE”, this situation shows Picture 7.



Picture 7: Example – step 3

Finally we connect objects “SCIE” and objects “NCE”. By it, we constructed a Pile structure representing string “SCIENCE”. See Picture 8.



Picture 8: Example – step 4

All objects in this structure are the same type. The operations associated with this type allow generate the string or substring of the data represented by the objects. [1]

Take note that for each symbol there is only one object representing it, and each time, when string already occurs, then this objects is reused. Once created, many times reused.

Applications:

Pile is extremely good for [2]:

- Totally flexible databases allowing instant arbitrary views (every field is key field)
- Integration of all applications
- Cognitive applications (complex, adaptive, learning)
- Overcoming current complexity barriers
- Overcoming current data explosion barriers
- Mapping of any describably or traceable structure
- Overcoming current incompatibilities of standards, formats, types
- Overcoming the dependency on Moore's Law
- A new relationist computer science

Cognitive library

Because PILE works with connections which are referable objects, it seems to be predestined for the programming of cognitive libraries and semantic networks. What PILE does in creating a new connection is very alike to what happens in our nervous system as thinking (and also learning): one state of the nervous system is connected with another state, then a third state can be connected with this connection etc. [1]

Bioinformatics

Maintaining a very fast substring search (exact as well as non-exact search), which is the first and simplest application of PILE, promises immediate profit for bioinformatics applications. These applications include:

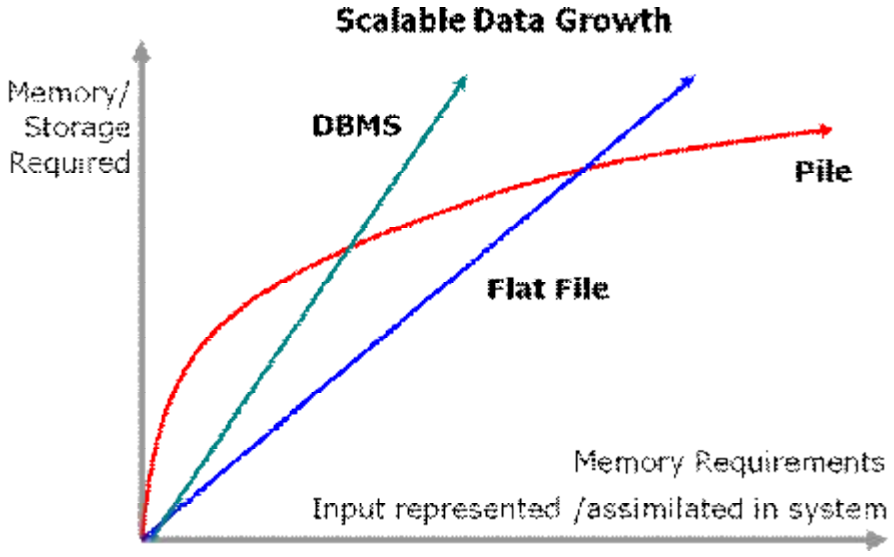
- searching in many databases at once (which can be semantically connected in PILE)
- sequence alignment (comparing two DNA or protein sequences and "fitting" them to other)
- multiple sequence alignment – a number of sequences have to be "fitted" to others at once
- clustering of ESTs (pieces of genes, which have been sequenced and have to be ordered and placed in the "big" genome)

In general, clustering algorithms are a further field of application of PILE. [1]

Databases

For any kind of databases we expect a great benefit from the use of PILE. The access to the data is no longer limited to the key/field access. The search for any entry/part of an entry is equally fast. The query format doesn't have to be a key/field entry; it can be any part of an entry, a combination of entries or of parts of entries. Especially we get access from the content of a cell to its key/field entry or from a child in a hierarchical structure to its parent. The size of the query is unlimited. For big databases and for similar data samples the compression effect of PILE also occurs.

An important issue of PILE is that there is no data redundancy. This implies that the procedure of updating a database is fast, easy and produces no additional errors. [1]



Picture 9: Data growth id databases [2]

Others

These possibilities open new horizons for search applications from hard disk scanning on a separate computer to internet search engines.

Many other application areas require trans-contextual connection and dynamic data generation. More prominent among them are computational linguistics, compression schemes, data communication (not only over LAN and WAN, but also between nodes in parallel systems and clusters). Since the PILE approach automatically creates a fully integrated and connected data space, application areas like Data Mining and Knowledge Management can benefit. PILE also holds promise for interactive and environmentally sensitive systems such as robotics, machine control and simulation systems. Last but not least PILE opens new possibilities for AI applications, cognitive computing, machine learning, Complex Adaptive Systems (CAS) and other areas where scalability due to data sizes or the inability to map complex systems have hindered the development of large scale applications. [1]

Literature:

[1] KRIEG P. Assimilative Computing: A Radical Relationist Approach. [online], 2005, [cit. 3.4.2007], <http://www.knowledgeboard.com/download/2774/Assimilative_Computing.ppt>.

[2] MATURANA H. Biology cognition. Biological Computer Laboratory Research Report BCL 9.0., Urbana IL: University of Illionois, 1970.

[1] Pile system Inc [on-line]. ©2000-2006, [cit. 3.4.2007], <http://www.pilesys.com>.

[4] PROUTSKOVA P. The Pile System: A New Approach To Data and Computing. [online], 2004, [cit. 3.4.2007], < http://www.pilesys.com/new/documents/Pile%20Math%20Intro.pdf>.

This paper wrote with respect of Erez Elul, inventor of Pile technology, and Pile systems Inc., company developing this technology. Thank for big inspiration.

Contact adress:

Ing. Milan Tomeš
University of Pardubice
Fakulta ekonomicko-správní
Ústav systémového inženýrství a informatiky
Studentska 84
53210 Pardubice
Email: milan.tomes@upce.cz
tel. č.: +420 466 036 147