DESIGN PATTERN FOR FUZZY AND OBJECT ORIENTED DATABASES[*]

O. Volráb

Czech University of Life Sciences, Faculty of Economics and Management, Department of Information Engineering, Prague, Czech Republic

One of the frequently used practices for sharing of software solutions is a design pattern form. It is a widely accepted method describing the answers for recurrent issues during a software development process. This article shows the way how to utilize a design pattern form for fuzzy approach in object oriented database systems. It introduces an original pattern as an effort to provide an easy understandable and general solution which is not tightly coupled with any particular database system and/or programming environment. There is also an overview of some reference implementations appended at the end as a proof of applicability in real-life situations.

design pattern; fuzzy; sets; OODBMS; object database; methodology, pattern

INTRODUCTION

During the last few years there have been done many research works in fuzzy logic scientific field. Similarly, an area of the object oriented databases (OODBMS) has reached utilization in many situations during information system (IS) development.

The utilization of fuzzy approach within a design and implementation of database oriented information system is however limited by tools and techniques which are available to regular IT staff. Therefore it is constantly necessary to overrule barriers between research results and their applications in real life. For this purposes there are some well known forms accepted by majority of IT community and general understandable. A design pattern method (Ambler, 1998; Fowler, 2002; Gamma et al., 1995) is such a form which suits well for general and versatile solutions of software development. Other well-known forms are cookbooks or recipes but these are usually tightly coupled with particular programming environment.

Motivation for problem solution

Different methods for fuzzy approach in an OODBMS have been already introduces by number of authors. However, existing works are focused on a concrete database platform, interface for persistence and/or programming language. A question concerning this situation rises with an aspiration:

- To unify and integrate existing methods.
- To remove close dependencies on technologies.
- To formulate a solution in understandable form of design pattern.

MATERIAL AND METHODS

A design pattern how to implement fuzzy approach to OODBMS is a result of research in many scientific fields like construction of OODBMS and fuzzy logic theory. With respect of this article purposes a classic GoF pattern style (Gamma et al., 1995) is slightly shorten and modified.

RESULT – DESIGN PATTERN FOR FUZZY OODBMS

Intent

A pattern introduces a fuzzy set concept into an OODBMS interface. A fuzzy set is based on definition of membership function and entry elements specification. It provides an interface for basic fuzzy set operations by means of general strategies and allows changing of partial algorithms independently.

Motivation

Information system development (regulation or business type) with utilization of fuzzy logic approach for decision making is very often coupled with processing of large data stored in databases. A character of these data is usually highly structured with many relationships which come from complex technological or economical processes. In case that some OODBMS is used for data storing then it is necessary to find some useful and object oriented method for assigning data to fuzzy sets. These sets are later used in common IF-THEN rules.

* Supported by the Ministry of Education, Youth and Sports of the Czech Republic (Grant No. MSM 6046070904 – Information and knowledge support of strategic control and No. 2C06004 – Information and knowledge management – IZMAN).

The more complex a process is the bigger demands on fuzzy set construction and manipulation exist. With respect to relative lower standardisation in OODBMS area (in opposite to RDBMS) it is also necessary to deal with different implementations for every single database engine. To the most important aspects which should be addressed belongs:

- Where and how to represent fuzzy set itself. Mass data processing should be expected and for this reason an appropriate mechanism of fuzzy set storage proposed. Data volume influences also the implementation way of fuzzy set operations which cannot be limited by size of operational memory.
- Context sensitive mechanism for selecting appropriate type of fuzzy operators. How to provide standard types of operators but still have an opportunity to add custom implementations.
- Membership functions can vary depending on a type of decision process. It means that more than one membership function can exist for particular data at the same moment. Therefore, definition of membership function should not be static.
- Information about degree of memberships should not be a part of element itself. This enables to use any object stored in database as an element of fuzzy sets.
- Changes in stored data, membership functions and algorithms of fuzzy operations should not be difficult to accomplish because in such a complex data structures they are likely to happen very often.
- Fuzzy extensions of OODBMS may benefit from specific features of particular database system but with respect to general structure and behaviour to keep a software solution portable and reusable.

Applicability

Design pattern for fuzzy OODBMS is useful to apply in following situations:

- There is a significant amount of data stored in an OODBMS and we would like to classify these data into the fuzzy sets according to our criteria. At the same time we do not want change data in any way (to keep data independent on fuzzy processing).
- When it is useful to have the same data model for fuzzy sets in application logic on client side and in a database system on server side. In such a situation we can move the components of software solution independently on data processing.
- The algorithms of membership evaluation and fuzzy operations are context dependent. This pattern allows distinguishing of different client requests at the same time (parallel processing in different contexts).
- It is essential to be able to move created information system with fuzzy approach to a different software platform. Changing of OODBMS should not require changing a data model.

Table 1. Reference implementation matrix

Platform/OODBMS	Java 5.0	Smalltalk	C# .NET
Versant	X		
Gemstone/S		X	
Objectstore	X		
db4objects			X

Structure

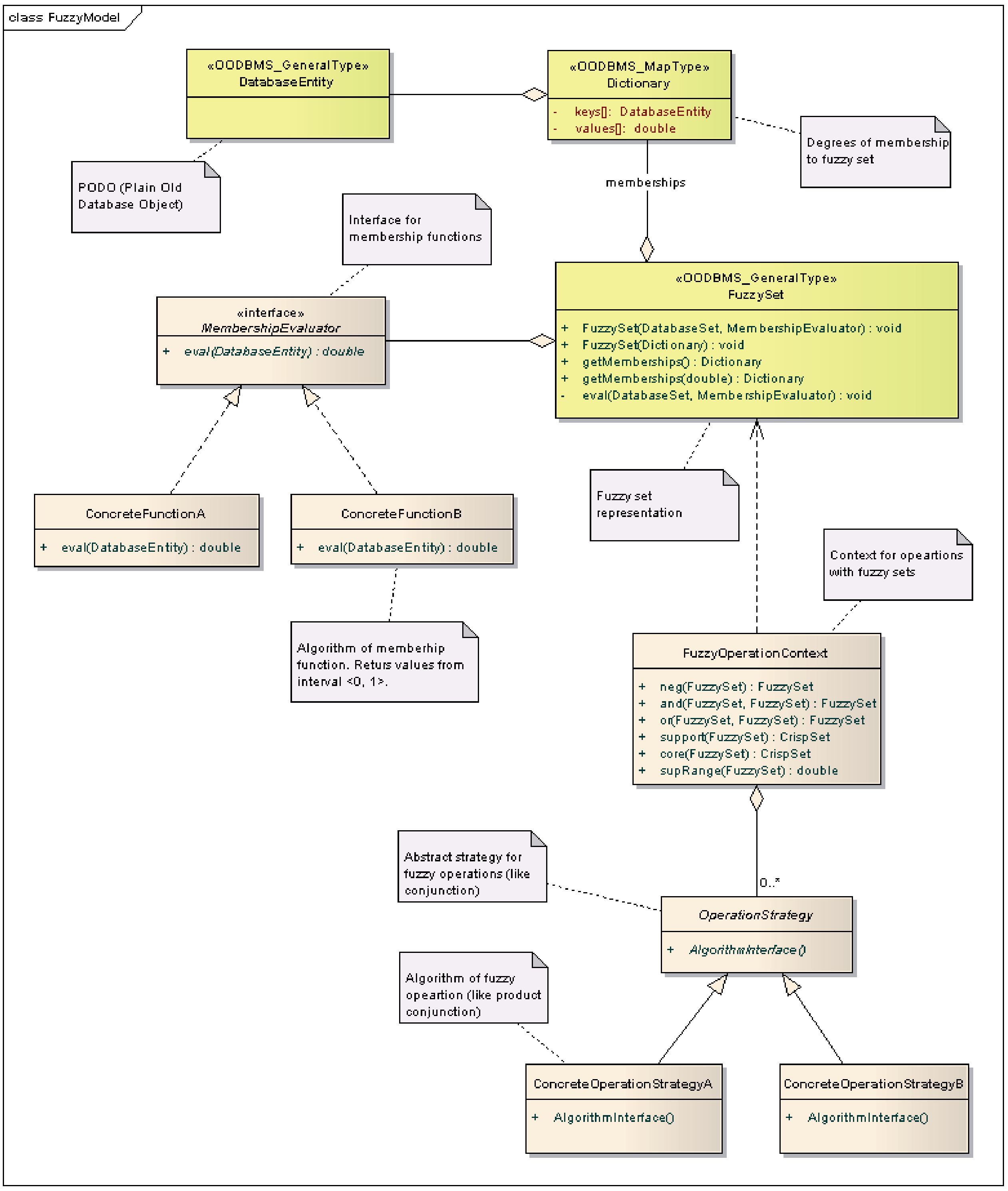
class FuzzyModel
«OODBMS_GeneralType» DatabaseEntity
«OODBMS_MapType» Dictionary
- keys[]: DatabaseEntity
- values[]: double
Degrees of membership to fuzzy set
PODO (Plain Old Database Object)
memberships
Interface for membership functions
«interface» MembershipEvaluator
+ eval(DatabaseEntity) : double
«OODBMS_GeneralType» FuzzySet
+ FuzzySet(DatabaseSet, MembershipEvaluator) : void
+ FuzzySet(Dictionary) : void
+ getMemberships() : Dictionary
+ getMemberships(double) : Dictionary
- eval(DatabaseSet, MembershipEvaluator) : void
ConcreteFunctionA
+ eval(DatabaseEntity) : double
ConcreteFunctionB
+ eval(DatabaseEntity) : double
Fuzzy set representation
Context for opeartions with fuzzy sets
Algorithm of memberhip function. Returs values from interval <0, 1>.
FuzzyOperationContext
+ neg(FuzzySet) : FuzzySet
+ and(FuzzySet, FuzzySet) : FuzzySet
+ or(FuzzySet, FuzzySet) : FuzzySet
+ support(FuzzySet) : CrispSet
+ core(FuzzySet) : CrispSet
+ supRange(FuzzySet) : double
Abstract strategy for fuzzy operations (like conjunction)
0..*
OperationStrategy
+ AlgorithmInterface()
Algorithm of fuzzy opeartion (like product conjunction)
ConcreteOperationStrategyA
+ AlgorithmInterface()
ConcreteOperationStrategyB
+ AlgorithmInterface()

Fig. 1. The design pattern structure

Participants

- FuzzySet
  - The fuzzy set representation in client (application logic) and in database server. All FuzzySet objects are stored in OODBMS (possibility to leave fuzzy sets transient is also available).
  - It is configurable via ConcreteFunction object which is used for assigning the membership degrees.
- Dictionary (OODBMS_MapType)
  - A definition of internal database structure for fuzzy set elements and their membership degrees. If possible, some kind of database map (data types derived from ODMG3 standard) is used.

- DatabaseEntity (OODBMS_GeneralType)
  - Any object stored in a database with a capability to become a member of fuzzy set.
- MembershipEvaluator
  - A declaration of simple interface for algorithms of membership functions.
- ConcreteFunction
  - An implementation of concrete algorithm of membership function. Computed membership degrees are transferred to fuzzy set via MembershipEvaluator interface.
- FuzzyOperationContext
  - A logic of fuzzy set operations dependent on current query context. For each operation with multiple implementations (variants) there is a reference to OperationStrategy object.
- OperationStrategy
  - An interface declaration (as an abstract class) for implementation of fuzzy set operation. It provides an independent access to real fuzzy operations (details of implementations are encapsulated).
- ConcreteOperationStrategy
  - An implementation of particular variant of fuzzy operation declared in OperationStrategy interface (see Strategy pattern in GoF – G a m m a et al., 1995).

Consequences

When using a design pattern for fuzzy OODBMS it is recommended to take into account the following advantages and disadvantages:

1. *Fuzzy sets stored in a database.* When a database is used as storage for fuzzy sets then there is no limitation of client operational memory for maximum set size. There are only limits of database system itself. On the other hand, fuzzy sets are still created dynamically with query context sensitive parameters. This representation of fuzzy set does not make use of wrapper mechanism (Adapter or Decorator – G a m m a et al., 1995) for each element but uses instead a shared data structure of map type to keep all membership degrees of one fuzzy set together.
2. *Independency on specific database interface.* The pattern is applicable along with various database interfaces. It can be implemented with some of the standards like JPA (Java Persistence API), ADO.NET as well as with native database interfaces that offer specific performance and semantic advantages. In some types of OODBMS the pattern can be used for implementation of application logic directly in the database itself.
3. *Context oriented membership functions.* A simple interface for membership algorithms allows changing of their definitions from the client system according to the current query needs. This is extremely useful when two parallel queries with the same data but different membership functions are requested at the same time.
4. *An encapsulation of implementation details.* The pattern is coupled with a classic Strategy pattern (G a m m a et al., 1995) that is used for encapsulation of fuzzy operations details. A client system selects appropriate strategy in accordance with time and space complexity and/or query needs. A relation between operation and its algorithm stays dynamic so that client system can combine different implementations within the same query (e.g. standard fuzzy conjunction for outer condition and product fuzzy conjunction for inner condition).
5. *No modification of primary data.* Fuzzy extension in the pattern can be used for any object stored in a database. There is no need to implement any specific interface or make any other changes. The author of this article suggested an acronym PODO (Plain Old Database Object) for this kind of data stored in OODBMS. The name was inspired by well-known acronym POJO from M. Fowler, R. Parsons and J. MacKenzie, which is used for simple Java objects. The only requirement for database objects is a public interface that is used in membership function.
6. *Caching the results.* From the time and space complexity point of view it is very useful sometimes to store partial results of fuzzy queries. This is realized through the storing of FuzzySet instances directly in a database. When it is reasonable a cache mechanism can be extended beyond the scope of one database transaction. In such a case it would be necessary to handle correctly concurrent updates of membership degrees as well as adding and removing elements from fuzzy sets. This is not covered by the pattern.
7. *Strict object orientation.* An interface for fuzzy approach in OODBMS keeps simple object oriented way and does not need for its implementation in programming languages any special text-oriented constructs (contrary of the fuzzy approach in M a et al., 2004; M a , 2005). An object orientation of all pattern design makes easy dynamic work with programming code like refactoring or application of agile methods.
8. *Portability between OODBMS.* There are only three database dependent classes in the pattern: DatabaseEntity, Dictionary and FuzzySet. These classes use only the very basic data structures and even if they use platform specific names they can be substituted. The instances of DatabaseEntity class are simple PODO objects – no special requirements. Instances of Dictionary class are implementations of any general data map. The key class FuzzySet needs only to accept realization of MembershipEvaluator interface. If there is no capability of OODBMS to use an interface it can be transparently substituted by abstract class.
9. *A limitation of results optimization.* The pattern has potential disadvantage in query optimization. The FuzzySet class accepts a reference to database query result and fetches the result itself. In some cases there is some possibility to tune up the result fetching by database specific parameters (bulk loading, cache optimization, proxy handling). In such situations the pattern might be extended. An extra strategy for result fetching can be implemented and then used by FuzzySet

class. A responsibility for the best fetching strategy would be on client system side. This disadvantage is automatically eliminated if membership function can be evaluated directly in OODBMS but only some database systems are capable of this.
10. *Interface oriented extensions.* The fuzzy extension in design pattern is proposed on interface level. It means that no changes are made in OODBMS architecture and logic. This can be a source of lower optimization than native database implementations can achieve. On the other side solutions based on this pattern are more reusable and portable.
11. *Number of objects.* A combination with Strategy pattern comes with a higher number of objects in a data model. A common solution is outlined in G a m m a et al. (1995). In case of single implementation of each fuzzy operation it is possible to merge responsibilities of ConcreteOperationStrategy and FuzzyOperationContext classes.

Reference implementations

Design patterns are usually based on long experience with recurrent situations. These situations need general but customizable solutions. The "Design pattern for fuzzy OODBMS" has been developed the very same way. It is based on fuzzy set theory, fuzzy logic and experience with many situations from the platforms presented in Table 1.

There is a summary of reference implementations on different OODBMS platforms and programming languages in Table 1. All the implementations use the same general data structure in accordance with the design pattern. There were no restrictions in any of the class-instance OODBMS or object oriented languages. In addition, some implementations have utilized their specific features like generic data types, object blocks, native queries, clustering technique (V i s n i c k , 2003) and others.

CONCLUSION

The design pattern introduced in this article has outlined the way how to overrule barrier between results from fuzzy logic research and their utilization in real database oriented IS. The pattern is applicable virtually in any class-instance OODBMS under stated conditions. At the same time, there is no limitation for object oriented programming environment as far as it is capable of connecting to OODBMS. In order to get feedback from real life situations some reference implementations have been created and tested. Results proofed the concept.

REFERENCES

AMBLER, W. S.: Process Patterns: Building Large-Scale Systems Using Object Technology. Cambridge University Press 1998.

FOWLER, M.: Patterns of Enterprise Application Architecture. Addison-Wesley 2002.

GAMMA, E. – HELM, R. – JOHNSON, R., et al.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley 1995.

MA, Z. M. et al.: Extending Object-Oriented Databases for Fuzzy Information Modeling. Information Systems, 29, 2004 (5): 421–435.

MA, ZONGMIN: Advances in Fuzzy Object-Oriented Databases: Modeling and Application. Idea Group, Hershey 2005.

VISNICK, L.: Clustering Techniques in Objectstore. Technical White Paper. Retrieved September 2003 from http://www.objectstore.net.

Received for publication on January 14, 2008
Accepted for publication on March 3, 2008

VOLRÁB, O. (Česká zemědělská univerzita, Fakulta provozně ekonomická, katedra informačního inženýrství, Praha, Česká republika):

Návrhový vzor pro fuzzy a objektově orientované databáze.

Scientia Agric. Bohem., *39*, 2008: 77–81.

Jedním z často používaných způsobů sdílení myšlenek softwarových řešení je forma návrhových vzorů. Jedná se o široce přijímanou metodu popisující odpovědi na opakující se problémy vznikající během vývoje softwaru. Tento článek ukazuje, jak je možné využít návrhové vzory pro rozšíření objektově orientovaných databázových systémů o možnosti nabízející fuzzy přístupy. Představuje původní vzor jakožto snahu o poskytnutí snadného, srozumitelného a obecného řešení, které není pevně spojeno s určitým databázový systémem a/nebo programovacím prostředím. V závěru článku je přidán přehled současných referenčních implementací, které mohou sloužit jako verifikace využitelnosti vzoru v reálných situacích.

návrhový vzor; fuzzy; množiny; OODBMS; objektová databáze; metodika; vzor

Contact Address:

Ing. Ondřej V o l r á b , Česká zemědělská univerzita v Praze, Fakulta provozně ekonomická, katedra informačního inženýrství, Kamýcká 1076, 165 21 Praha 6-Suchdol, Česká republika, e-mail: volrab@pef.czu.cz