

AID: an Object-Relational Schema Design-Tool

Hassan Badir¹, Etienne Pichat, He Xiaojun

LIRIS Laboratory, UFR/Info, UCBL, Lyon
8 Niel Bohr 69622, Villeurbanne, France
{hbadir}@liris.cnrs.fr

Abstract. Schema design is a complex activity that makes use of a variety of knowledge. It should take into account foreseen processing. Nowadays, interaction of processing with data (static) is not taken into account by our AID tool that is limited to offer a graphical interface to conceive in various models (UML Class diagrams, Universal relationship and object attributes forest), and transform the input diagram into the two other models. This tool highlights the complementarities of these different models, especially object attributes forest able to take into account designer knowledge of the processing.

Keywords. Object attributes Forest, Universal relationship with inclusions, Class diagrams, stereotype, SQL3, XML.

1 Introduction

Database design based-tools aim essentially modeling, schema conceptual design, transforming these schemas into implementation structures, with an optimization and querying aim. Nowadays, (Entity-relationship [6], UML diagrams [16], relational data model, Universal relationship with inclusions [18]...), database design methods (Object-Merise [19], UML, ...) and platforms (Super [10], Tramis [12], Power Designer [7], ConceptBase [13], Rational Rose [17], etc.) are widely available in the computer science community, by the way of published papers, and books or softwares. However, despite all these various tools; actually database applications are not designed in an automatic manner, not well documented, not well structured, and incoherent and not easy to maintain. Numbers of models and methods are proposed to the public, but insufficient to answer all their needs. Commercial Design tools are often suitable for a specific field and inadequate for the others or are not well-used.

In the framework of object information systems design, the aim of this paper is to present an advanced visual graphical environment and sufficiently rich semantically, that allows express non-expert user needs. Conceptually, it relies on object attributes forest constructed with an interactive help. This expressive representation is a set of attributes trees where leaves refer to a tree. We then transform object attributes forest into a

database stereotyped UML class diagram which uses UML-DB profile [3] [15] completed in [4].

This paper is structured as follows: section 2 discuss our work with respect to our global project, section 3 treat general principals of the used model Object Attributes Forest, section 4 discuss the derivation of class diagrams from object attributes forest. We conclude with an overview of the main contributions of this paper, followed by the future perspectives.

2 AID Architecture

In order to achieve the expected objectives of our work, we integrate three formalisms representing three different models of our tool called AID. AID is an edition and object, relational, and object-relational conception-assisted platform. Its roles can be enumerated as follows:

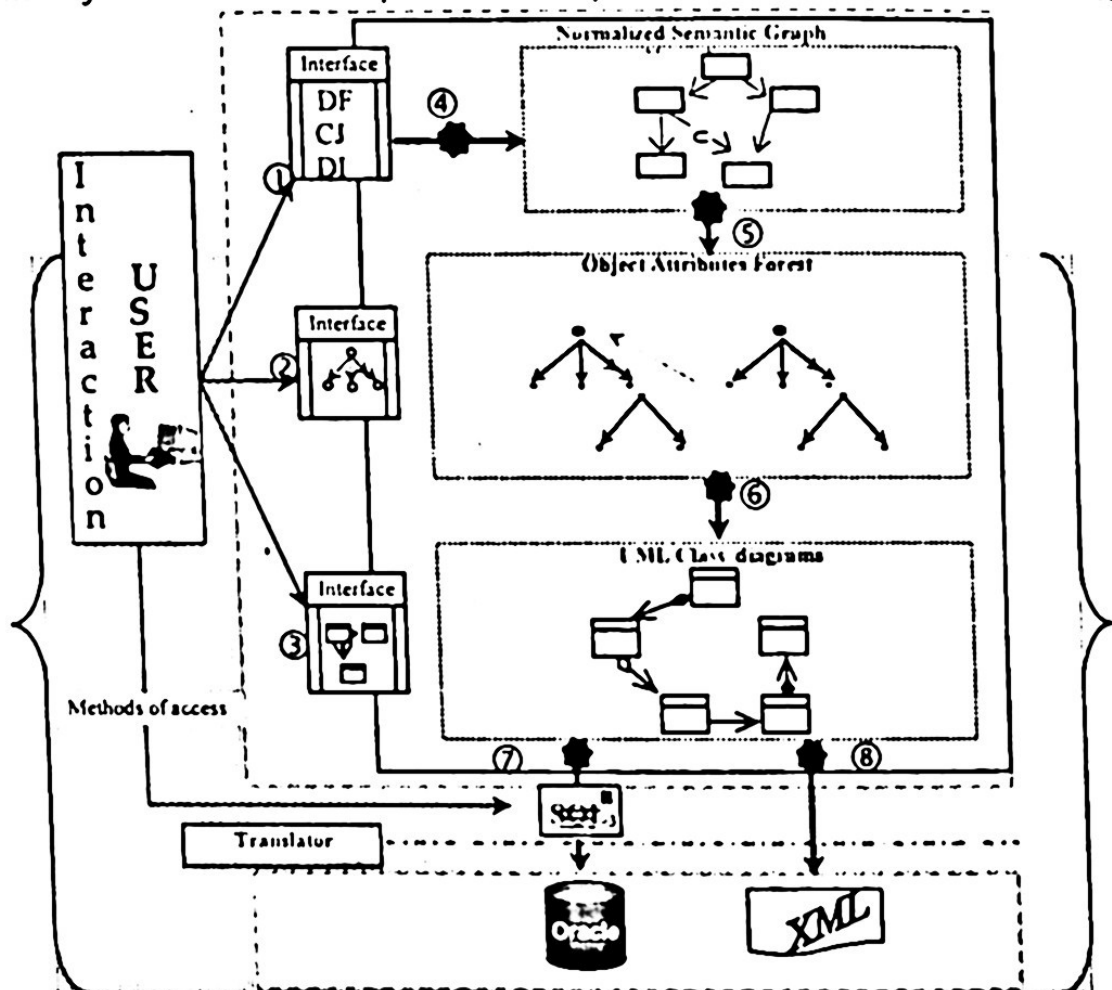


Fig. 1. AID architecture

- User can describe schema of his choice through on the following models (Fig. 1): Universal relationship with inclusions (URI) [18], object attribute forest and DB stereotyped UML classes.
- to go from a model to another by using transformation algorithms, it allow as well SQL3 description generation or XML schema,
- User can normalize; transform a representation into relationships related by inclusion dependencies in a Normalized Semantic Graph (NSG). This normalization could be partial, and keep unchanged some complex object structures fixed by user without any further normalization.
- To optimize the obtained conceptual schema with respect to the foreseen processing, introducing hence access methods, even to denormalize it.

This tool is composed of three windows or principales interfaces. Each window is reserved for representing different formalism. It is constituted of five modules of transformation:

- Interactive interface for specifying normalized semantic graph by applying a normalization algorithm on a set of functional dependences, jointure composantes, and inclusions dependences (Fig. 1)(1).
- Interactive interface to specify object attributes forest allowing describe complex data (Fig. 1)(2).
- Interactive interface for specifying database stereotyped UML class diagrams (3).
- Transformation module of an ordered normalized semantic graph (ONSG)[14] into an object attribute forest. The corresponding algorithm is given by [14] (Fig. 1)(5).
- Transformation module (Fig. 1)(6) of object attribute forest into database stereotyped UML class diagrams. The corresponding algorithm is described in section 4.
- Module for generating SQL3 description from a UML classes diagram (Fig. 1)(7).
- Module for generating XML schema from a UML classes diagrams (Fig. 1)(8).

In this paper, we will restrict our study at the shaded part which is composed of two interfaces ((Fig. 1)(2 and 3) and a transformation module ((Fig. 1)(6 and 7).

3 Object Attributes Forest (OAF)







The main advantage of the relational model is its great simplicity in representing data in the form of relation (table). In this model, the schema of a relation is composed of a set of attributes taking values from atomic domains: it is the first normal form (1NF). However, such an inherent constraint requires serious limitations in terms of modeling. In order to overpass such a limit, a model with complex values (structured values, or complex objects), very simple, has been introduced to propose to user a maximum of possibilities in designing his reality. This model uses the least set of modeling operators. It's in this objective that has been used OAF, conceptualization of complex object [7][2].

3.1 Elements composition Object Attributes Forest

Elements composing Object Attributes Forest are attributes tree and reference edge. Recall that a tree is a non-cycled oriented graph having one and only one root (or node without predecessor). Its other nodes called intermediates if they have at least one successor and leaf otherwise.

Attributes tree is a tree where each node is has a named attribute. The root name is also the tree name and will be the name of the associated class. Each intermediate node or root is unshared and its lifetime is of the composed class. Each arc of attributes tree is:

- Univoque (total or partial): to each value of its original attribute (node) is associated at most a value of its extremity attribute. It's represented by a simple arrow, or.
- Multivoque (total or partial): to each value of its original attribute could be associated a set of values to its extremity attribute. It's represented by an arc with a double arrow in the same direction (table. 1)

| Table 1. Representation of the bows of a object of attributes forest | | | | | |
|--|--|--|--|--|--|
|  |  |  |  |  |  |
| Univoque (1,1) - (1,n) | set (0,1) - (1,n) | Multivoque (1,n) - (1,n) | set (0,n) - (1,n) | Reference (heritage) | Inclusions dependences |

A OAF is a set of attributes trees and reference arcs. A reference arc relates a leaf node to an attributes tree. It indicate that each value of leaf attribute is a reference (a pointer) to at most one instance or value of the root of the referenced attributes tree. It is represented by discontinued fat arrow.

The naming of the attributes forest nodes should verify the following constraints:

- The name of the attributes tree should be unique among the attributes trees names.
- The node name should be unique among the nodes names of the same immediate predecessor.

(Fig.2) represent an object attribute forest of the Olympic-Games modeling athletes, coaches, and sportive events.

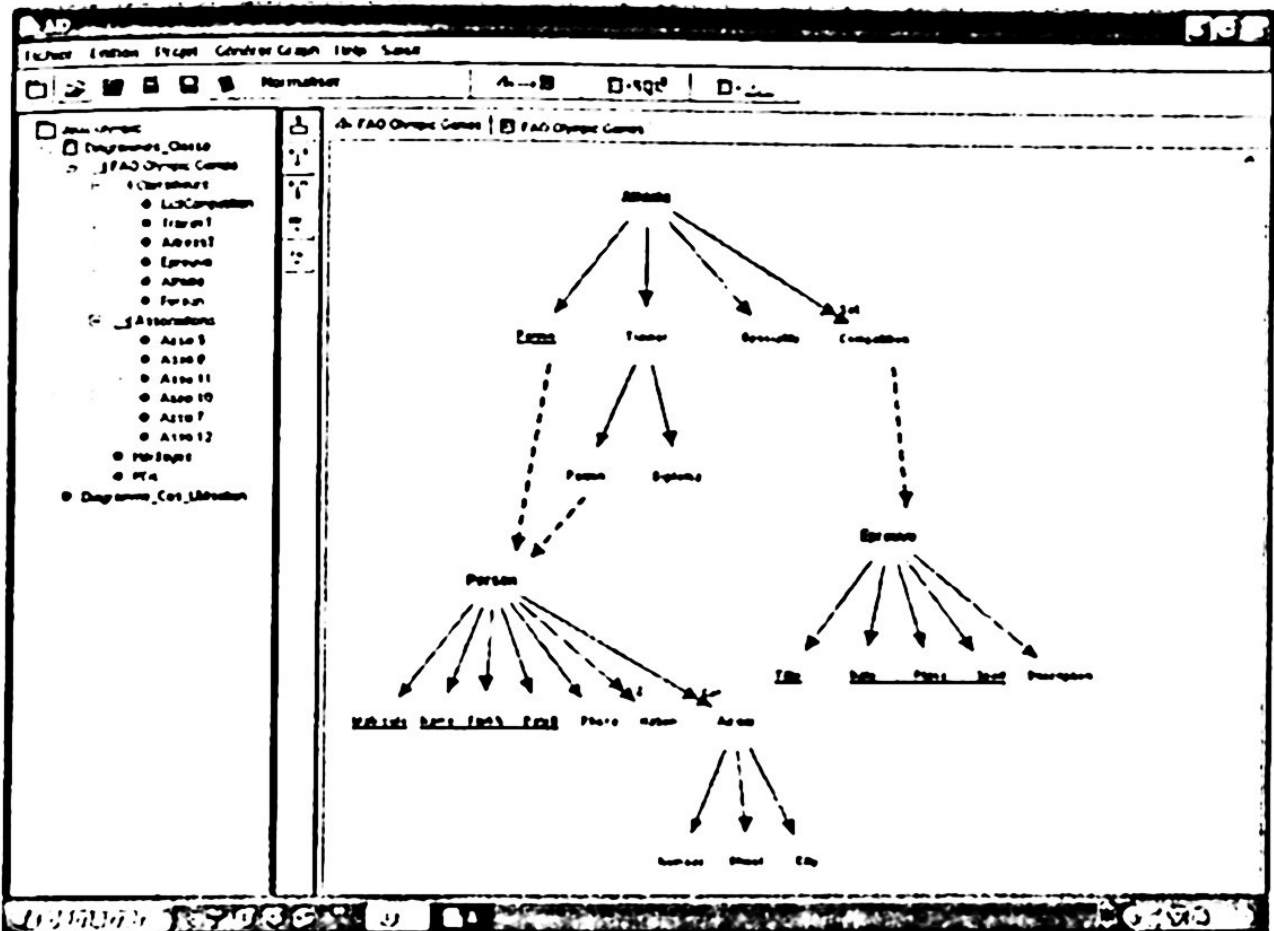


Fig. 2. OAF of Olympic-games where for Person the second key Name FirstN Tel not displayed

In an object attribute forest, each attributes tree has a color chosen hazardly by AID or clarified by user. This color will be inherited by UML class diagram in the transformation phase.

3.2 Keys in a OAF

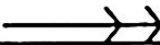
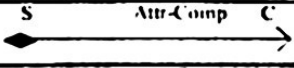
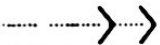
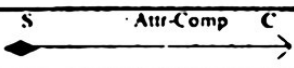
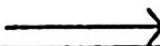
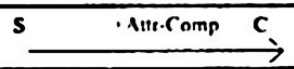

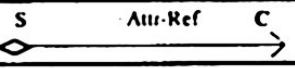
Object identifier in object data modeling is provided at the same time of the object creation. It is not often visible to users (or applications). Object identity is insufficient. It's important to give users the possibility to access objects using their contents and to avoid double insertion (equal objects or having the same key value). The key concept is an essential semantic concept in data modeling, introduced later by object DBMS. [5] A complex class or attributes tree could have one or several keys. For example (Fig. 2), Person has as keys *Matricule*, *First-Name Second-Name BirthDate* and *First-Name Second-Name Phone-Number*.

4 Database Stereotyped UML Schema Derivation from a Complex Tree

This section relates complex data representation and UML class diagram, which represent the cornerstone of UML method. Recall that a class diagram is composed of classes and relations between these classes, with their attributes and methods. We give at first a algorithm for transforming object attributes forest into classes diagram, then after some explanations about the used UML-DB profile, we explain how stereotypes are generated from attributes tree keys.

4.1 Algorithm for transforming OAF into classes diagram

This algorithm (Algo.1) constructs persistent classes from roots and intermediate nodes of OAF, adds to them attributes from leaves, and defines compositions from arcs and associations from references. It highlights the recapitulated correspondences (Table 2):

| Table 2 Correspondences between arcs of OAF and associations of UML classes diagram | | |
|---|---|---|
| Types | Arcs | Relations |
| Illimited multivoque composed |  |  |
| Borned multivoque composed |  |  |
| Univoque composed |  |  |
| Reference [univoque] |  |  |

Algorithm 1: Algorithm of transformation of a OAF in a Classes Diagram

Input : Object Attributes Forest.

Output : DB Stereotyped UML classes Diagram.

For All each tree (of root R) **do**

If its class (or those of it root) R doesn't exist **Then**

 CreateClass(R)

End If

For All Arc(S_o, S_e) in an order that predecessor has been examined **do**

If (S_o, S_e) is univoque **Then**

If S_e is intermediate **Then**

 CreateClass(S_e)

 Compose univoquely this class to class S_o .

End If


```

If  $S_e$  is a leaf and reference the tree  $S$  Then
If referenced class doesn't exist Then
    CreateClass( $S$ )
End If
    Associate referenced class  $S$  to class  $S_o$  with role  $S_e$ 
End If
If  $S_e$  is a leaf not referenced Then
     $\text{Attr}(S_o) = \text{Attr}(S_o) \cup \{S_e\}$  {Associate attribute  $S_e$  to
    classe  $S_o$ }
End If
Else
If  $(S_o, S_e)$  is multivoque Then
    CreateClass( $TasS_e$ )
    Compose multivoquely this class to classe  $S_o$ 
If  $S_e$  is a leaf and referenced  $S$  Then
If referenced class doesn't exist Then
    CreateClass( $S$ )
End If
    Associate referenced class  $S$  to classe  $TasS_e$  with
    role  $S_e$ 
End If
If  $S_e$  is a leaf not referenced Then
    {Associate attribute  $S_e$ } to class  $TasS_e$ 
End If
If  $S_e$  is intermediate Then
    CreateClass( $TasS_e$ )
    Compose multivoquely this class to classe  $S_o$ 
End If
End If
End If
End If
End If

```

Algorithm execution applied to OAF of (table 2) gives the UML class of (Fig.3). Therefore, the node *Address*, which has an entering arc (*Person*, *Address*) of labeled multivalued NT type, generate between the two classes *Person* and *Address* the composition arrow into *TasAddress* taking the +*Address* role *TasAddress* side, the competition of the going node of reference type, generate the composition arrow of Competition into *TasCompetition* and aggregation arrow of *TasCompetition* into *Epreuve* having the +*Competition* role side *Epreuve*.

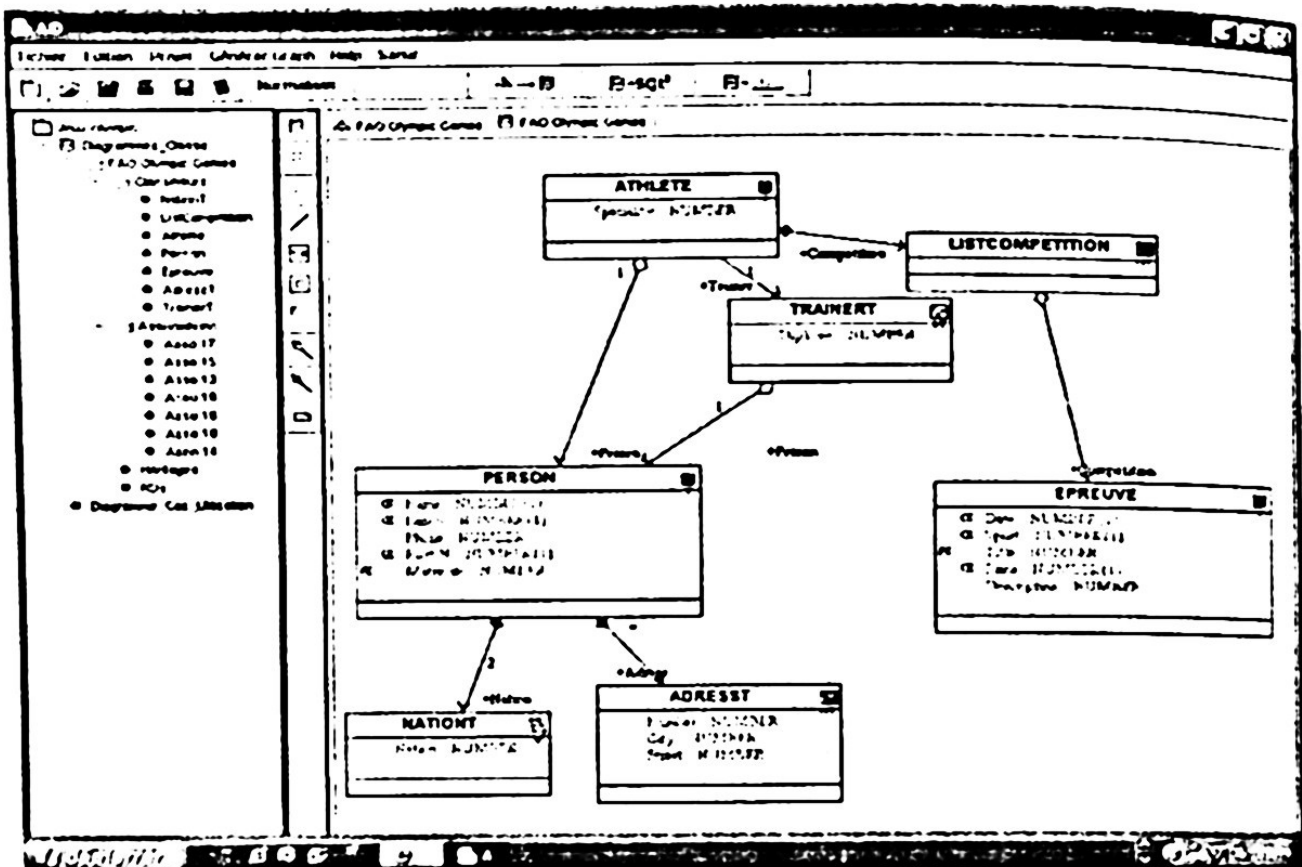


Fig. 3. DB stereotyped classes diagram obtained from The FAO of the (Fig 2)

4.2 UML-DB profile

The profile concept is essential in model engineering. This concept is not neutral; it descends principally from *hypergenericity* [9] concept. The profile concept has been standardized in UML 1.4 in 2000 and reinforced in UML 2.0. A profile is a UML extension that keeps intact the UML metamodel [17]. It's composed of a set of stereotypes, of labeled values attached to these stereotypes and constraints (table 3). In the field of databases, we have extended the UML-DB [4] profile to composed or simple candidate keys. This profile has also icons stereotypes representing nested table, array, type and table.

4.3 Keys processing

To transform keys of objects attributes forest into keys in classes diagram, we have used UML-DB profile for primary, secondary, simple, and composed keys specifications. We have used stereotypes to express keys in classes diagram. In the design phase, we can associate several keys to one class, and each key can contain several attributes. One of these keys could be specified as primary: we stereotype it as PK as we can see it in (fig 4).

Stereotyping the others keys often called candidates, is more difficult. Each key is identified by an index that will be associated with its attributes. Thus, each candidate key is stereotyped CK, and each one of its attributes carries on its index as showed in (fig 4). This stereotyping generalizes that adopted by Rose [17].

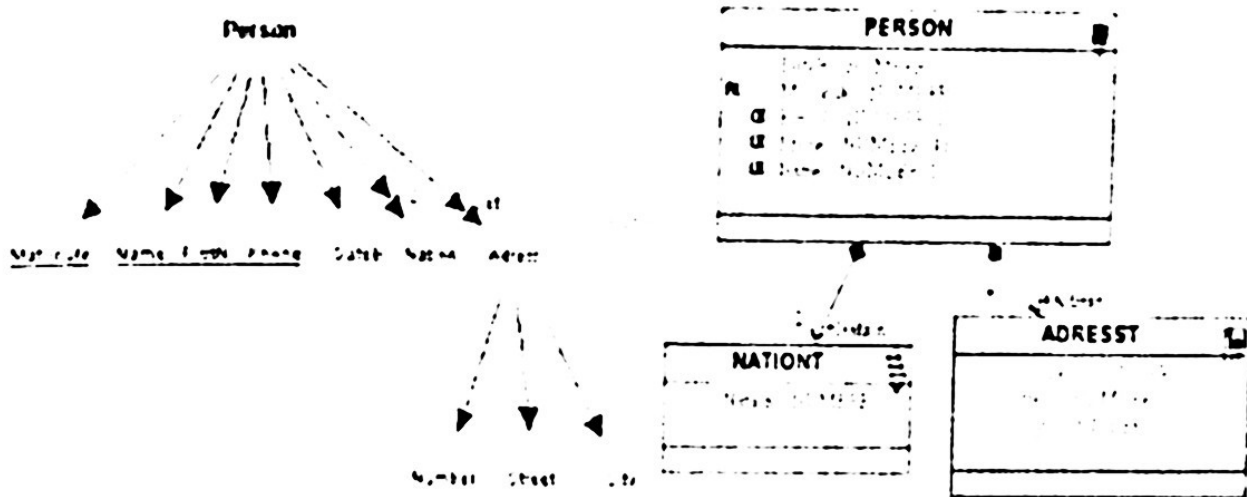


Fig. 4. Computing and transforming complex keys 3

The following table resumes a part of the UML-DB profile with stereotypes for Oracle9i:

| Stereotype | Icon | Description | Constraint |
|------------------|------|---|---|
| <<Object-type>> | | Allows the representation of new user defined data types. It corresponds to the structured type in SQL:1999 | It can only be used to define value types |
| <<Object-table>> | | It represents a class of the database schema that should be defined as a table of an object type. It corresponds to the typed table in SQL:1999 | A typed table implies the definition of a structured type, which is the type of the table |
| <<Nested-Table>> | | Represents an non-indexed and unbounded collection type | The elements of a NT can be of any data type except another collection type |
| <<Varray>> | | Represents an indexed and bounded collection type. Its corresponds to the array type in SQL:1999. | The elements of a VARRAY can be of any data type except another collection type |

Table 3. UML-DB Profile for Oracle9i

Algorithm 2 Algorithm of transformation of keys

```

For All each root  $S_0$  do
  For All each arc  $(S_0; S_e)$  do
    If  $(S_0; S_e)$  is univoque then
      If  $S_e$  is leaf not refernced then
        If  $S_e$  2 an Primary Key PK then
          Stéréotype  $S_e$  = «PK»
        End If
      For  $j = 1$  to  $\text{card}(\text{setofCandidatesKey})$  do
        If  $S_e$  2 an Primary Key PK then
          Stéréotype  $S_e$  = Stéréotype  $S_e$  + «PK»
        Else
          Stéréotype  $S_e$  = «CK»
          Contrainte  $S_e$  = Contrainte  $S_e \cup \{j\}$ 
        End If
      End For
    End If
  End For
End For

```

In an objects attributes forest, keys could be represented in the same time with PK, CK, and index symbols. Such a representation seems to us less visual. Thus, we have adopted the underlining like entity-association diagrams, that supposes doubling attributes belonging to several keys.

5 Conclusions

In our paper, we have addressed quality improvement in designing complexes relational-object database schemas and the construction of UML classes diagram from an objects attributes forest. We have presented an algorithm for transforming a complex tree into a UML DB stereotyped classes diagram for conceiving object-relational schema. Our choice for using objects attributes forest as conceptual data model is justified by its simplicity in representing complex data, and data modeling. The enrichment, by keys and references that we have added, leads to a better complex data design. In other hand, we have chosen as a final conceptual model, UML class diagram thanks to its expressive semantic expressiveness, and its standardization, after extending it in order to support object-relational and relational database design. We have developed an intuitive interface for assisting design. The goal is to make the conception process very simple and very easy for non-specialists users. Thanks to this interface, user conceive his OAF progressively. While he refines his schema, user can eventually come back, and go in other direction.

Interface offers flexibility and conviviality to express graphically trees, and inheritance management by its references. It will be very promising to derive object-relational schemas from relational schemas. Other possible objective is to extend our tool for automatic transformation of relational model (Normalized Semantic Graph) into an object-relational model (UML class diagram). Thus to improve our tool (processing, offering a graphical query interface for precise specifications).

References

1. Abitboul S., Hull R., Vianu V., Foundations of Databases, Addison-Wesley, 1995.
2. Abitboul S., Hull R., Buneman P., Suciu D., Data on the Web, Addison-Wesley, 1999.
3. Ambler S., Persistence a UML Profile for Data Modeling, www.agiledata.org, 2002.
4. Badir H., Pichat E., Beqqali O., Utilisation d'UML pour concevoir une base de données objet ou objet-relationnelle IEEE SETIT, 2003.
5. Cattell R.G.G., Skeen J., The Object Database Standard, Morgan Kaufmann Publishers, 1994, ACM Transaction.
6. Chen P. P.: The Entity-Relationship-Model-Toward a unified view of data, ACM Transaction on Database System 1 (1), 1976.
7. DATE C.J., An Introduction to Database Systems, Editions Addison Wesley, 2000.
8. Dellobel C. Bancilhon R., Building an Object-Oriented Database System- The story of O2, Morgan Kaufmann Publishers, 1992.
9. Desfray P., Object Engineering, The fourth dimension, Addison Wesley 1994.
10. Dennebouy Y., Andersson M., Auddino A., Dupont Y., Fontana E., Gentile M., and Spaccapietra S, SUPER: visual interfaces for object + relationship data models, Journal of visual languages and computing, 6(1):27 -52, 1995.
11. Fallouh F., Données complexes et relation Universelle avec inclusions. Une aide à la conception à l'interrogation des bases de données, thèse de doctorat de l'université UCB Lyon1, 1994.
12. Ilick J-M, Englebert V., Henrard J., Roland D., Hainaut L.-L., The DB-Main Data-Base Engineering CASE Tool (Version 6) - Functions Overview, Technical manual, Institut d'Informatique, FUNDP, November 2000.
13. Jarke M., Gallersdörfer R., Jeusfeld M.A., Staudt M., Eherer S., ConceptBase – a deductive object base for meta data management, In Journal of IIS, Vol. 4, No. 2, 1998
14. Manea A., Towards an object Oriented Design Method. Application to Hydrological Database, DEXA 1995: London, United Kingdom.
15. Marcos E., Vela B. and Cavero J. M., Extending UML for Database Design, Fourth International Conference on the Unified Modeling Language: UML 2001, Toronto, Canada.
16. Muller P.A., Gaertner N., Modélisation objet avec UML, Eyrolles, 2e édition, 2000.
17. Naiburg E. J., Maksimchuk R. A., UML for Database Design, Addison-Wesley, 2001.
18. Pichat E., Bodin R., Ingénierie des données, Masson, 1990.
19. Rochfeld A., Rigaux P., Traité de modélisation objet, Editions Eyrolles, 2002.