The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.32 e-ISSN: 2251-7545

An Ameliorated Methodology to Establish Compatibility between Structural Parts of Object Oriented Technology and Network DBMS.

Siddappa G Makanur
Dept. of Information Science & Engineering
STJ Institute of Technology,
Ranebennur, India
sgmakanur@hotmail.com

Dr. Shivanand M. Handigund
Dept. of Computer Science & Engineering
Bangalore Institute of Technology,
Bangalore, India
smhandigund@gmail.com

Dr. M. Sreenivasa Rao
School of Information Technology,
JN Technological University,
Hyderabad, India
srmeda@gmail.com

Abstract-- **Object Oriented (OO) concept is widely accepted for software development by the software development community for its naturalness and mathematical rigor. Object Oriented Analysis & Design (OOAD) is supported by Unified Modeling Language (UML). Though, OO Technology (OOT) is developed with the state of the art technology, it is passive i.e. it can not be implemented for the development of software projects on its own. On the other hand the Network Database Management System (NDBMS) is active for the implementation of the information system, but suffers from lack of au-courant technology. Both are having complementing characteristics of each other. This paper identifies these complementing virtues and lacunae of both the paradigms and super imposes one over the other. The super imposed paradigm nullifies lacunae of one with the virtues of the other and vice versa. This superimposition is used to develop a robust object network database management system. This act necessitates establishment of compatibility amongst model elements of OOT and NDBMS. In this paper, we have developed an ameliorated methodology that brings compatibility between these two paradigms. This transforms the OOT paradigm from passive to active at the same time it provides NDBMS with state of the art coating without enhancing its complexity. We have mapped model elements viz. Class to Record Types, inter relationships such as Association, Composition and Aggregation to Set Types, inheritance of superclass and subclasses to record type, at present this is achieved at the cost of introducing a constraint that subclasses are non overlapping**.

KEYWORDS: Modeling elements, UML, OOAD, OOT, CODASYL, NDBMS, Bachman diagram, ORDBMS, OODBMS, Multiple Inheritance.

I. INTRODUCTION

Motivation: There exists many DBMSs like Hierarchical DBMS, Network DBMS, RDBMS, ORDBMS and OODBMS. But no DBMS ossifies all types of information systems. Each one may well suits some applications and ill-suits for other applications. This is because each DBMS compromises with some of the features for ease. Researchers expected that ORDBMS would be the ultimate. Unfortunately it has its own lacunae in implementing nonconventional interrelationships like aggregation, generalization/specialization and multiple inheritance through the required hierarchical path etc. Moreover, the information system is realized through the DBMS with heterogeneous group of people with different cultural and language background. In such heterogeneous group there exists the possible use of synonymous and heteronymous words. There is no provision to resolve these synonyms/heteronyms issues present in DBMSs except Network DBMS. No DBMSs could realize effectively the multiple inheritance features. The different associations are realized in DBMS with OIDs (object identity) like Database Keys (DBK). On the other hand though the object oriented technology is the state of the art technology, it has precluded the implementation of good database design & good software engineering principles where as even in classical data processing systems these principles realized. There is an urgent need to blend these good principles into OOT components and then use them to refine NDBMS features.

Vision: To incorporate the au-courant OO Technology facilities into the Network database Management System.

Mission: To identify & attune the Object Oriented Technology features to blend into NDBMS without compromising the vigour.

Objectives:

1. To identify the structural model elements of OOT to be amenable for NDBMS model elements.
2. To establish the compatibility between OOT model elements and NDBMS implementation elements.

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.32 e-ISSN: 2251-7545

Object Oriented Technology is the state of the art technology, imbibed with naturalness and sound mathematical rigor. OOT is immortal to the extent of naturalness i.e. any thing developed on OOT will not be legacy in near future. Thus the NDBMS blended with OOT features will be robust and will not be legacy in near future. However, it is passive in nature which can not be realized independently. On the other side most of the characteristics of NDBMS are either compatible or achievable compatibility with the characteristics of OOT [8], therefore blending the two paradigms is quite justifiable.

The good database design principles are realized even in classical paradigms, but are not incorporated in OOT. This is reduction-ad-absurdum to be called as state of the art technology. Therefore blending of the virtues of good database design principles & good software engineering principles is need of the hour. Moreover passiveness of OOT is to be transformed into activeness.

On the other side the NDBMS is developed by CODASYL (COnference on DAta SYstems Languages), so the metonym is CODASYL DBMS. It has OOT compatible features like the Set Type (multiple inheritance), Hierarchical representation of attributes (superclass/subclass), Realm (Class/Use case package diagram), Location modes (file/memory organization), Navigation (access strategies), Insertion/Retention modes (constraints), authorization (access constraints) through Schema and Subschema definition & Privacy Locks, use of Alias clause (synonyms /metonym), User Work Area (program variables corresponding to database object classes), System Owned Set (database entry points) [3,7,13,14,15]. Thus it was developed as a highly active DBMS, unfortunately the navigation of the record occurrences through their hierarchical root node made it difficult to use with its high complexity.

NDBMS does not support data oriented process and the interrelationships among the record types are realized using set types, and between the record occurrences by the set occurrences. The order of record occurrences are stored through the DBK not trough the primary key and the foreign key. Though, the interrelationship is limited to an association, the naming of set types helps in incorporating composition/shared aggregation and generalization/specialization. Thus NDBMS is more suitable to realize OO features [8, 14, 15].

The only hitch in NDBMS is its complexity which can not be reduced in the ensuing ONDBMS and therefore the complexity wimp can be implicitly achieved through incorporating more facilities for the complexity (buy one get two). Reducing the complexity of NDBMS is a herculean task however we attempt to provide more facilities and sophistication so that the rate of complexity per facility or sophistication is drastically reduced.

The NDBMS has powerful implementation techniques but with complex schema and subschema definitions (uncomfortable bogies). The OOT has luxurious state of the art development model (comfortable bogies) but without implementation engine. Thus OOT and NDBMS complements each other (made for each other) to design and develop information systems, and this NDBMS is the powerful engine to drag the comforts of OOT.

In our intended work, we have mapped (establishment of compatibility) the model elements (structural) viz. Class to Record Type, interrelationships such as Association, Composite or shared Aggregation to Set Types with proper constraints such as retention/insertion criteria and mode, inheritance of superclass and non overlapping subclasses to Record Type with redefines clause. The private and public visibility is mapped to Privacy Locks [6]. The NDBMS has the facility to introduce user specific synonyms and heteronyms without conscious knowledge of the generic name stored in the schema [6, 13].

II. PROPOSED WORK

A. *Mapping OO features to compatible features of NDBMS*

In our intended work we consider the abstraction of subset of the model elements responsible for representing the structural part of the object. Further, these abstracted model elements are mapped to NDBMS acceptable features of the model elements. This mapping eases the process of implementing the corresponding OO features in an object network approach.

B. Mapping Class to Record Type

The concept of the class is based on the entity definition, and the record type in NDBMS also represents the entity. Object structure is similar to normalized entity structure [1]. The Class name is mapped directly to the Record Type name. The attribute of the class is mapped to data item of the record type. Since the record type is already designed using good

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.32 e-ISSN: 2251-7545

database design principles (normalized), while mapping class to record type, care should be taken to refine the class using good database design principles [1] so as to smoothen the mapping from class structure to a record structure. The attributes of record type need not be in the same level. These hierarchical level attributes may be used to represent the composite attributes in the class structure. Using usage clause the appropriate attribute are mapped efficiently (saves space). While defining the record type, its storage location modes (makes access criteria) are also specified for the speedy retrieval from the database.

Ex: struct USNType{
short Region;
char College[2];
short Year;
char Branch[2];
short Sn;
}

Class STUDENT {USN USNType;
NAME String;
Subj[8] String;}

Class is mapped to Record Type as below

```
RECORD NAME IS STUDENT.
LOCATION MODE CALC USING USNType.
01  STUDENT
  02 USNType.
    03 Region        PIC 99.
    03 College       PIC AA.
    03 Yr            PIC 99.
    03 Branch        PIC AA.
    03 Sn            PIC 999.
  02 Name            PIC X(25).
  02 Subj            PIC X(20) OCCURS 8 TIMES
```

C. Mapping Association to Set Type.

Association is the navigability or reachability between the related object classes. In NDBMS, the association is represented by set type. Navigation is made through these sets occurrences. An association is implemented directly as set type (one : many) in which class on *one* side is represented as the owner and class on *many* side is represented as member of the set type. No need of any pseudo attribute or method to represent an association [19] unlike in conventional relational models. The degree of participation is defined using retention clause in the set type definition. Association properties such as order can also be defined using order clause in set type definition. The set modes are defined to provide easy traversal to members. Association with cardinality m: n is implemented using two set types, in one set owner is record type at m side and in the other set owner is record type at n side. The relationship record containing primary keys of both the owner record types (can be dummy, need not contain any data) is member in both the set types and acts as a junction record.

D. Mapping Composition (whole-part relation) to set type

The composition is a tighter form of association, represented by a set type in which whole class is represented by owner record type and part class is represented by member record type, with retention mode as mandatory (fig1). So that part object can not exist in isolation from the whole and belongs to only one assembly and when whole is deleted, its parts are also deleted (same life time). Thus the semantics of composition is enforced directly. To facilitate better access, the location mode of part record type is defined as via set type. Thus composition is directly implemented as shown in the fig1.

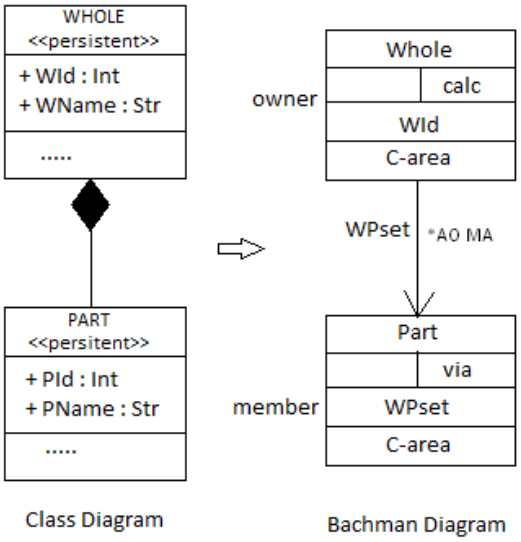

Fig1: Composition

The implementation schema definition for the composition (fig1) in CODASYL is as follows

```
RECORD NAME IS WHOLE
LOCATION MODE CALC USING WId
   DUPLICATES NOT ALLOWED FOR WId
WITHIN C-Area
```

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.32 e-ISSN: 2251-7545

```
03 WId          PIC 9(5)
03 WName        PIC X(20)

RECORD NAME IS PART
LOCATION MODE IS VIA WPset
WITHIN AREA C-Area
03 PId                  PIC 9(5)
03 PName                PIC X(20)

SET NAME WPset
ORDER NEXT
OWNER WHOLE MEMBER PART
MODE PONTER ARRAY
MANDATORY AUTOMATIC
SET SELECTION IS THRU LOCATION MODE OF
OWNER //reach the set through its owner, as PART is weak
entity
```

When we read a composite object, it is essential that its parts would be read along. In DML(Data Manipulation Language) we can read all the PARTs at a time i.e. read all the DB Keys of the members in the set occurrence (Ex: WPset). To facilitate this, the mode of the set type should be POINTER ARRAY, then using ACQUIRE(20, List1, SET='WPset') verb all the DBKs of parts are accessed. This facilitates the semantics of a composite object when it is loaded into the host program (persistent closure is maintained).

E. Mapping Aggregation to Set Type

The aggregation is a form of association called has-a type of relation similar to composition but part is an independent object and can be shared by many assemblies (other aggregate object). This is an instance of many: many association and represented by two set types (1:m and 1:n) with the junction containing the primary keys of both record types or dummy record type as member in both the set types. One set type owned by an aggregate record type and other set type owned by part record type (fig2). Here the mode of both the set types should be ARRAYPOINTER so that the programmer can read all the parts of an aggregate object at once, and the semantics of aggregation can be maintained easily.

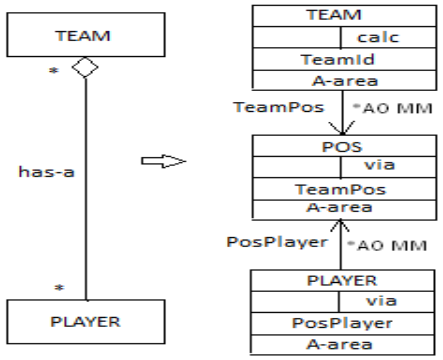

Fig2: Aggregation

F. Mapping Generalization/specialization.

In the database the objects which are the instances of the class are stored. The instance of a class in the inheritance tree contains the attributes which are the union of attributes of all its superclasses with its own attributes. Thus it is appropriate to merge the classes in the in the single inheritance tree. But the subclasses at the same level are mutually exclusive and hence the instance contains the attributes of any one subclass, attributes of other classes are not applicable, this leads to the wastage of space. This wastage is eliminated by allocating the same space for the attributes of all subclass in the same level. This is accomplished by the use of Redefines clause available in DDL (Data Definition Language) of CODASYL model.

The inheritance tree is represented by single record type by grouping the attributes in the hierarchical object classes from root node to leaf nodes in higher level numbers (lower levels) in schema and subschema definitions. We represent the disjoint portion of subclasses at the same level in the hierarchical tree through the use of Redefines clause at the same level. The overlapping attributes of the subclasses can be represented by the same level on par with redefines clause immediately prior to redefines clause. This serves the purpose of representing superclass subclass hierarchy but at the cost of increase in memory space. Moreover the classes at the same level need to be represented with the same level number, this optimizes the use of memory location, but the subclasses of the same level can not be concurrently processed (fig3). This technique is superior to single table mapping technique in object-relational mapping, where the inapplicable attributes values are padded with null[18].

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.32 e-ISSN: 2251-7545

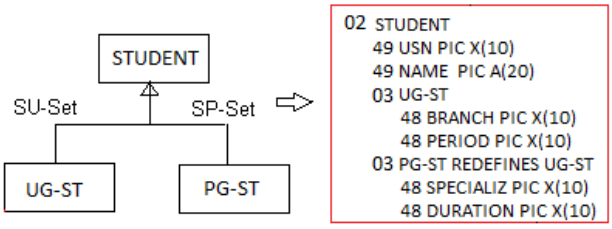

Fig3: Inheritance (disjoint subclasses)

In case of single inheritance tree, the private visibility is not useful as it contradicts the very purpose of superclass and subclass hierarchy. All the classes in the specific single inheritance tree are merged to root class, and the names of these classes becomes synonyms. This synonyms can be implemented using alias clause in the subschema definition.

G. Mapping Multiple inheritance:

In this case, we use the above said merging of classes to single class that is transformed & represented in a schema as a single record type with hierarchical level numbers for the single inheritance. For multiple inheritance part, we use the owner member relationship (set type) between super and subclasses, thus each super class is connected with the set type(fig4).

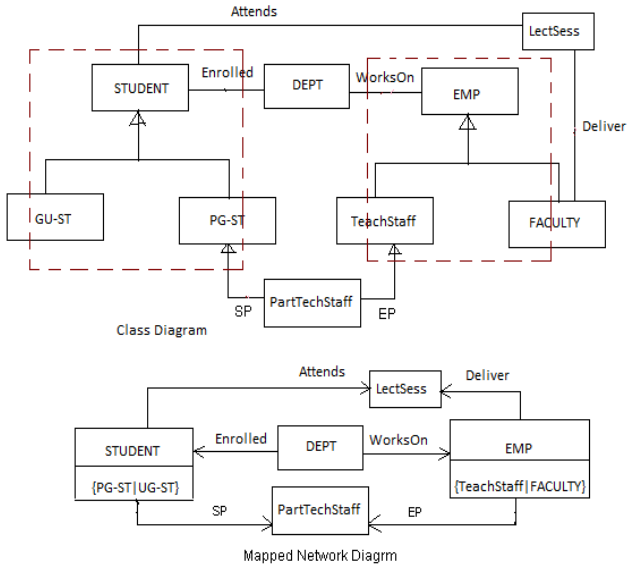

Fig4: Multiple Inheritances.

The multiply inherited record type is a member in every inherited path from each owner. Here, using the *virtual* clause the member record type has to share the required attributes through the each inheritance set type in which it is a member (i.e. it shares the attributes from the two set types SP and EP as shown in fig4).A typical member record definition is shown below.

RECORD TYPE IS PartTechStaff
02 Skill PIC X(25)
02 StName IS VIRTUAL AND SOURCE IS StName OF OWNER OF SP
02 StEmpID IS VIRTUAL AND SOURCE IS EmpID OF OWNER OF EP
Thus the programmer can share the required attributes from each owner (represented by the superclass) through the corresponding set type.

G. Mapping Diamond inheritance

This is similar to the above, but the classification is overlapping and redefines clause is not used for the single inheritance portion of the class hierarchy(fig5).

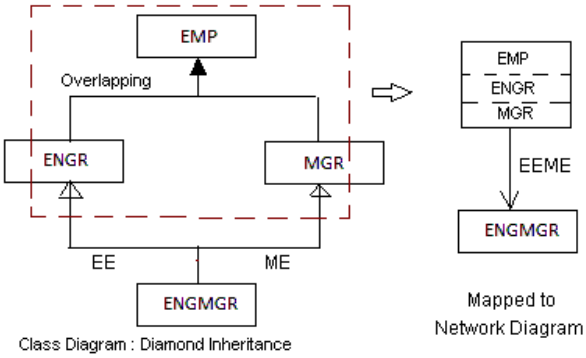

Fig5: Diamond Inheritance.

According to the representation of single inheritance, subclasses are merged with their superclass with redefines clause used for the attributes of the subclasses at the same level. But in diamond inheritance subclases are not mutually exclusive and hence memory for all attributes of all the subclasses along with base class attributes is required therefore Redefines clause should not be used. In fig5 the class ENGRMGR can avail any attributes of its superclasses after merging through EEME set type.

H. Visibility

Every host program is attached to a specific subschema. The record types and set types, which are constituent of subschema are only visible to the host program. Apart from this we can restrict the visibility at record level, attribute level, area level and set type level by using PRIVACY LOCKs defined in the subschema. While importing the

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.32 e-ISSN: 2251-7545

record type definition from schema in subschema, the PRIVACY LOCKs can be defined, viz. COPY recordtypeName RECORD PRIVACY LOCK FOR {DML Verb} IS {YES|NO}. And while importing set type definition like COPY setName SET PRIVACY LOCK FOR [set operation] IS {YES|NO}. Thus the private visibility is implemented using privacy locks.

Thus visibility of the record type is same as the visibility of an object class. The public visibility of record type is incorporated by the presence of required portion of record type description as part of the subschema and private with privacy locks. We have adopted this approach on the pretext of the utility of the attributes of the record type is restricted to adjacent (connect via an association or set type) record types through parameter passing.

III. RELATED WORK

A. *Relational approaches for the storage of objects.*

For the storage of objects, programmers are using RDBMS engines to implement information systems. Several Object Relational Mapping techniques and tools are being used by programmers' community, who spend 25% of their effort in managing their object to relational mappings and vice versa [5, 6, 10, 18] because of the semiotic difference between object modeling techniques and relational data modeling techniques.

In mapping classes to relations, when the class hierarchy is mapped to single relation. This mapping will produce nulls for the attributes that do not apply to the corresponding subclass[16, 18]. In our approach we have used the *redefines* clause in the single record definition to eliminate nulls. In ORDBMS each relation need an additional attribute to hold the tuple's unique identifier as OID (object identity)[16]. In our approach a database key acts as an OID and does not need any additional attribute

Thus the object oriented features are orthogonal to the relational model features and is not suitable to store objects [5]. Little analogy exists between the relational data model and the object data model, blending the two systems could not render the expected results [14, 15].

As per Atikinson et el, the CODASYL model supports certain OODBMS features defined in his first OODBMS manifesto[8] but has not discussed how it can support. An Object Database Management Group standards recommends that navigation should be supported by OODBMS [11] & CODASYL supports it. Therefore we are realizing the OO features into the compatible features of CODASYL DBMS to facilitate easy implementation.

Now days organizations are using ORDBMS to store their objects structure using Object-Relational Mapping[17, 18]. ORDBMS performs badly for single object operation or navigation using joins, which are slower than pointer traversal [2, 12]. The object oriented applications are not data centric and has broader range of relationships than those expressed in SQL [16].

B. *Object Oriented Database Management System (OODBMS):*

There is no truly portable way of interacting with an object oriented database exists and programmers are still managing with object-relational mappings [9]. Incompatibility arises as they uses different approaches to implement persistence[4]. The Object applications are not data centric like relational model, but OID centric [16]. NDBMS is also not a data centric no need of a foreign key and used DBKs to refer record objects. OODBMS aims at seamless integration with programming languages and NDBMS provides seamless integration by defining language specific subschema[6]. Object Management Group recommendations such as support for multi valued attribute, navigation, OID, language specific binding are already available in NDBMS. Thus enhancing the network data model is easier than developing a new OODBMS.

The structural CODASYL model for object oriented interrelationships was proposed by us in [14, 15]. This was based on incorporation of additional interrelationships such as aggregation, generalization/specialization as new kinds of set types. This may work well for isolated structural representation. This was wimp representation as there is a need to incorporate other features in a single representation. Here authors have tried to alter the CODASYL record organization itself instead of using the compatible features.

IV. CONCLUSION

In our paper, we have presented the techniques to map object class, various types of associations, generalization specialization and composite aggregation into Record Type, various types set types, along with their retention and insertion criteria respectively. Also we have developed

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.32 e-ISSN: 2251-7545

mapping techniques to realize visibility through privacy locks.

V. FUTURE WORK

We could not implement multiple inheritance in a clear cut way as it may take multiple routes to the navigation of parent classes. This we will take up as a challenge for our future work. While incorporating public visibility and protected visibility, there is a need of redundant records in every subschema and use of virtual attributes respectively. How to reduce this redundancy is yet be addressed.

REFERENCES

[1] Ajeet A C, Shivanand M H, " A Mediocre Approach to Syndicate the Attributes for a Class or Relation" , in proceedings of International Journal of Software Engineering and Its Applications, Vol. 5 No. 4, October, 2011.
[2] David Maier, Jacob Stein, Allen Otis, Alan Purdy. "Development of an Object-Oriented DBMS". OOPSLa'86 as ACM SIGPLAN 21, Nov 1986, 472-482.
[3] Elmasri and Navathe, "Database Systems" 2nd Edn, Pearson Education Publisher, 1994
[4] ETH Zurich's lecture "Object-Oriented Databases" by Michael Grossniklaus and Moira C. Norrie. http://www.globis.ethz.ch/education/oodb, Sept 2010.
[5] H. Darwen and C J Date, "The Third Manifesto", Technical Report, http://www.acm.org/sigmod/record/issues/9503/manifesto.ps , 1995.
[6] International Computers Ltd "IDMS Data Administration Languages", Reference., 1978.
[7] Jeffrey O Ullman, "Principles of Database Systems" 2nd Edn, Galgotia, 1984.
[8] Malkolm Atkinson et al, "The Object Oriented Database System Manifesto", First International Conference on Deductive and Object-Oriented Databases. Kyoto, Japan 1989.
[9] Michael C, "Next Generation Object Database Standardization", Object Database Technology Working Group White Paper, Sept 2007.
[10] Neal Leavitt, "Whatever Happened to Object-Oriented Databases?", IEEE Spectrum, Aug 2000
[11] Object Management Group, "The Object Data Standard: ODMG 3.0" , 1999. Morgan Kaufmann Publishers, San Francisco, California.
[12] R. G. G. Cattell, T. R. Rogrss. Combining Object Oriented and Relational Models of Data. 1986 International Works shop on Object-Oriented Database Systems, Pacific Grove, Calif, Sept 86.
[13] Robert W Tailor and Randall L F, "CODASYL Database Management System" Computing Surveys, Vol 8, No1, March 1976, ACM
[14] S. G. Makanur, Shivanand M. H. and Sreenivasa Rao, "An Ameliorated CODASYL Model to Store Object Structures" in proceedings of ICFoCS 2011, ISBN 978-81-921929-0-1, Aug 2011, paper serial no 59
[15] S. G. Makanur, Shivanand M. H. and Sreenivasa Rao, "Storage of Persistent Objects using Ameliorated CODASYL Model", inproceedings of IBM Centennial Colloquium and IBM Collaborative Academic Research Exchange (I-CARE) 2011, Oct 2011. Track III, Sl no: 06. URL: www-07.ibm.com/in/research/icare_2011_posters.htmlCached
[16] Thomas Connolly and Carolyn Begg, "Database Systems", IV Ed Peasrson. 2005
[17] W. Keller, "Persistence Options for Object Oriented Programs" European Conference on OO Programming Languages, Munich, 2004.
[18] Wolfgang Keller, "Mapping Objects to Tables A Pattern Language", Europian Pattern Languages of Programming Conference, Irsee, Germany, 1997.
[19] Yann G et at, "Bridging the gap between modeling and Programming language", Technical Report, Object Technology International, Inc. – 2670 Queens view Drive – Ottawa, Ontario, K2B 8K1 – Canada, 2002.