Transformation of Legacy System OODB Modeling into AODB

[1]Shivani A Trivedi, [2]Nalin. N. Jani, [3]Amal Kumar

[1]KSV-SKPIMCS-MCA, India,
[2]KSV-SKPIMCS-MCA, India,
[3]MBA-Bhavnagar University, India

***Abstract* - This paper aims at modeling database management system to support for effective intelligent decision making. The relational database (RDB) system has served for decades and provided relevant information to the needs of information seekers. The need for decision making met to transform RDB to ODB. This transformation could realize information for decision support. The dynamic situation is generating a need of decision making system rather decision support system. This need has inspired authors to model agent oriented databases (AODB). The paper has narrated the modeling and transformation from RDB model to ODB model to AODB model.**

***Keywords* — Data Modeling, Object Oriented data modeling, Agent, Agent Technology, Agent Oriented Database Model**

I. INTRODUCTION

The development of computerized application system started with applications in smallest domain as a FORTRAN, COBOL etc which was design and development that time procedural programming, the only procedural programming was available that span of time.

The advantage of language ‘C’ also supported regularly procedural oriented programming paradigm but added a domain of system software development. Then the extension of ‘C’ emerged as ‘C++’ good gave both object oriented paradigm.

The data were managed in the application systems but not scientifically the way the data was managed in a DBMS environment with availability of emerging development and resource reach execution platform such as J2EE and .NET framework based development. This development could lead the developers through Client Server architecture based application development with scientifically managed data through DBMS server. These applications could increase their domain from database server to web server giving flexibility to the clients to be connected to web database server on wider scale. These systems could add the features of object oriented programming in the development software for better manageability of S/W product. These systems classified under Object Oriented Database Management System (OODBMS) could give as robust applications but the object worked as they were predefined.

Today, the situation is that only organization using such s/w system faces certain key issues for their survival and growth of the organization with still better management of data, information and optimized decision in a chunk of time. Object Oriented Database Management System (OODBMS) do not come up to the mark to address these issues. The efforts was made to address these issues by developing component based database system, where intelligently objects were grouped and as a result it could reduced the time to take decision but the demand of organization still went high in a competitive environment. System should act intelligently and work for optimized decision, this lead development environment to the Agent Oriented Database Management System (AODBMS). These systems are capable to address issues such as reeducation time in decision making and making decision more intelligent. Concept of object is needs a transformation into the concept of Agent". An agent is a computer system situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives." An autonomous agent should be able to act without the direct intervention of humans (or other agents), and should have control over its own actions and internal state. And an "agent based system" means one in which the key abstraction used is that of an agent [1]. “Agent” is applied to diverse technologies and products.

II. RDBMS MODELING TO OODB MODELING

In RDB, the relationships among records are specified by attributes with matching values. These can be considered as value references and are specified via foreign key(s), which has a domain of primary key. These are limited to begin single-valued in each record because multi-valued attributes are not permitted in the basic relational model. Thus M:N relationships must be represented not directly, but as a separate relation (table) with application of 4th normal form as a part of normalization process.

In ODB, the relationships are typically handled by having relationship properties or reference attributes that include Object Identifiers, OID(s) of the related data objects. The role of primary key in RDB is played by OID in ODB. The additional component of role is to references functions/methods also. These can be considered as OID references to the related objects. Both single references, for individual instance of the objects and collections of references for prototyping of objects are facilitated. References for a binary relationship can be declared in a single direction, or in both directions, depending on the types of access expected. If declared in both directions, they may be specified as inverses of one another, thus enforcing the ODB equivalent of the relational referential integrity constraint.

In RDB, the mapping of the relationships are specified using 1:1, 1:M and M:N among the records. And the degree of relationship can be specified using unary, binary , ternary and n-ary. Where as when we are trying to map relationship from RDB to ODB the mapping of binary relationships that contain attributes is not that simple and the designer is required to select the direction of the attributes to be included. When the attributes are included in both directions, then redundancy in storage will occur and this may lead to inconsistent data. In this situation it is more preferable to use the relational approach of creating a separate table in RDB and creating a separate class in ODB. This approach can also be used for n-ary relationships, with degree n>2.

The handling of inheritance in RDB is by way of extended relationship and the same are represented graphically by way of Extended Entity Relationship Diagrams. The handling of inheritance in ODB, is by way of building structures into the model and the mapping is achieved by using the inheritance constructs, such as derived (:) and EXTENDS. In RDB design, no such built in constructs are facilitated for inheritance handling.

In RDB design it is not requirement of specification of operations during the design phase and they are required in the implementation phase. In ODB design, it is necessary to specify the operations during design phase since they are the part of class specifications.

Here we are trying to justify the conversion of RDB and ODB. Let us take the example to implement the four different cases to convert the EER model to ODB model. Here we have taken the EER diagram of standard University database.

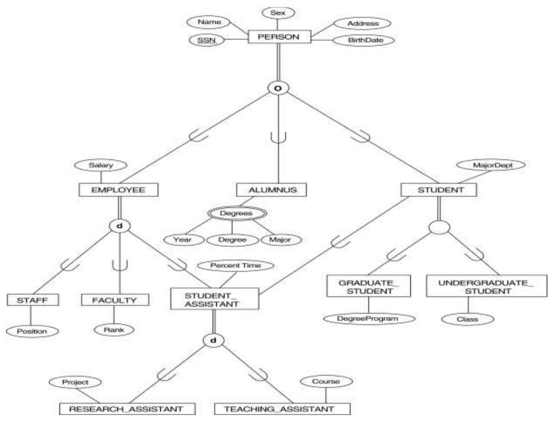

Fig. 1 Example of Extended ERD University Database in RDB

A. *Case 1: Entity type or subclasses*

Create an ODL *class* for each EER entity type or subclass.

- Multivalued Attributes can be handled as follows
  - List Constructor: attribute values are ordered
  - Bag Constructor: duplicates are allowed
  - Set Constructor: otherwise
- Composite Attributes can be handled using "Tuple Constructors".
- Key attributes can be specified as keys of extent
- Declare an extent for each class and specify any key attributes

Let us take the entity "PERSON" as shown in figure 1

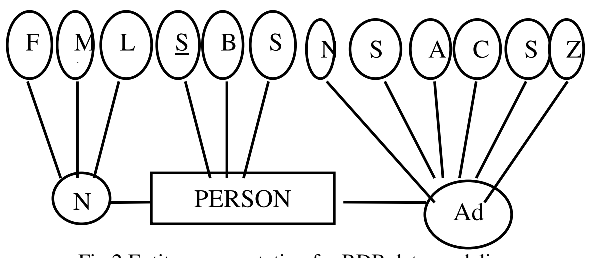

Fig 2 Entity representation for RDB data modeling

Let us draw a class diagram for the same as follow

```
PERSON
Key: Ssn
struct: Pname
          { string fname,
           string Mname,
           string Lname},
Struct: Address {short no,
                 string street,
                 short aptno,
                 string city,
                 string state,
                  short zip},
 date birthdate;
enum Gender {M, F} sex;
short age( );
```

Fig.3 Entity representation for ODB data modeling

B. *Case 2: Relationship*

I). Binary Relationship: We can add relationship properties or reference attributes into ODL classes that participate in the relationship, which references in both direction relationship properties that are inverses of one another or in only one direction attribute in the referencing class whose type is referenced class name

II). Cardinality Ratio: As discussed earlier in RDB,1:1 or N:1 and 1:N or M:N directions the relationship properties or reference attributes are identified as single-valued and set-valued or list-valued in ODB.

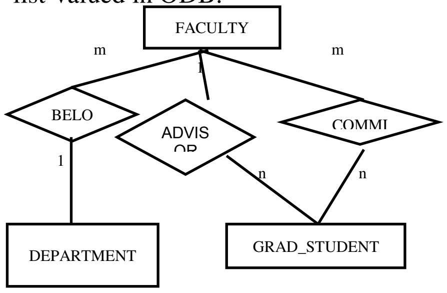

Fig. 3 An example of Binary relationship and cardinality ratio in data model of RDB.

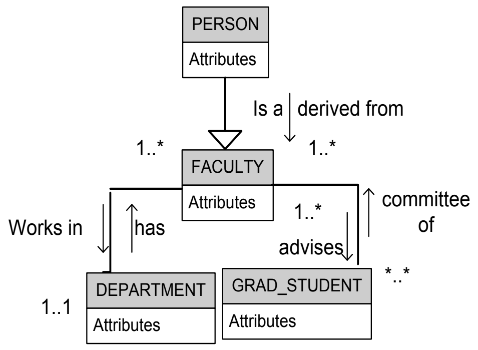

Fig.4 Transformation, an example of inheritance, single valued and set-valued data model of ODB.

C. *Case 3: Include appropriate operations for each class.*

I). Constructor method: check constraints that must hold when a new object is created.

II). Destructor method: check any constraints that may be violated when an object is deleted.

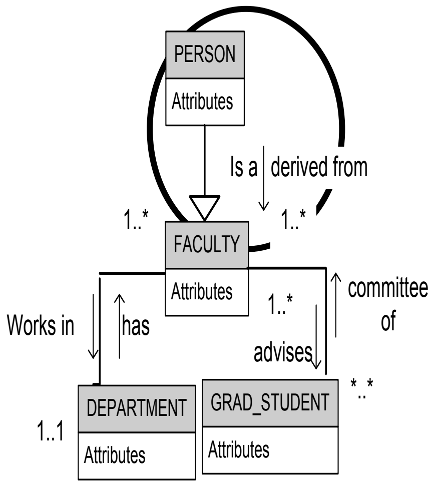

Fig. 5 PERSON is base class and FACULTY is derived class

D. *Case 4: An ODL class that corresponds to a subclass i.e faculty as shown in figure 5 can be described as*

```
class Faculty extends Person
(     extent   faculty)
{
     attribute string … ;
     relationship Department       works_in
                    inverse Department::has_faculty;
     relationship set<GradStudent> advises
                    inverse GradStudent::advisor;
     relationship set<GradSection> on_committee_of
                    inverse GradStudent::advisor;
void give_raise(in float raise);
void promote(in string new rank);
};
```

E. *Case 5: A weak entity type can be mapped the same way as regular entity type, if one: one relationship composite multivalued attributes of the owner entity type* set<struct<…>> or list<struct<…>>

III. NEED OF ODB TO AODB

To properly design an effective ODB, the database designer must have a clear understanding of the object-oriented model and its effective implementation through the model mapped prototype. Some data domains have explicit objects and clearly defined relationships among the objects. Such situations justify the application of object oriented principles where local integrity of data is guaranteed. Object-oriented Development is best suited for dynamic, interactive environments, as evidenced by its widespread acceptance in CAD/CAM and engineering design systems. Such systems have not proved their suitability to the same extent in case of enterprise wide applications that are dynamic, interactive and adaptive to changes in external environmental data. An ODB data model do contain system related definition/specification but needs in its extension the external environmental information for the intelligent decision support for dynamic decision driven business system.

In current scenario of business environment, practically it is not possible for decision makers to read every document that crosses their desks, every relevant data available in databases, every article in the magazine and journals to which they subscribe, or all most all the emails received in their mail boxes. This analysis is also supported by the survey findings of the Gartner group

- The amount of data collected by large enterprise doubles every year.
- Knowledge workers can analyze only 5% of this data.
- Most of their efforts are spent in trying to discover important patterns in the data i.e. more than 60%, a much smaller time is spent to determine that what these patterns mean, and very little time (10% or less) is spent actually doing something about the pattern.
- Tremendous of information reduces our decision making capability by 50%.

In spite of all of this, managers are expected to take account of key business information and make good decisions. In this situation "***Intelligent Agent***" is emerging as a suitable technical solution. A major value of ***intelligent agent*** is that they are able to assist in searching through large pool of data. They save time by making decisions relevant to the user. Information access and navigation are today's major applications of intelligent agents. Agents can handle many routine activities that need to be done quickly. In decision-making, intelligent agents can fulfill the growing needs for support for tasks performed by knowledge workers. The success of business in the market place can be achieved by the business professionals who can make timely and knowledgeable decisions; they can greatly increase their effectiveness by intelligently accessing information from the business databases. Database designers and developers are required to embed agent technology in managing the business databases. The embedded agent technology empowers the database environment for the support of dynamic parameter based intelligent decision making.

IV. AGENT TECHNOLOGY AN OVERVIEW

Three view point of an agent [2] "An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors."

"Autonomous agents are computational systems that inhabit some complex dynamic environment, sense and act autonomously in this environment, and by doing so realize a set of goals or tasks for which they are designed."

"Intelligent agents are software entities that carry out some set of operations on behalf of a user or another program with some degree of independence or autonomy, and in so doing, employ some knowledge or representation of the user's goals or desires."

In agent oriented paradigm agent can be defined like objects with additional component in its definition which is ***resources***. The ***autonomy*** characteristic of agent is referred as the freedom in the allocation of internal resource. The pro-activeness is another characteristic of an agent by which the agent is free to decide when to become active, deactivate and reactive.

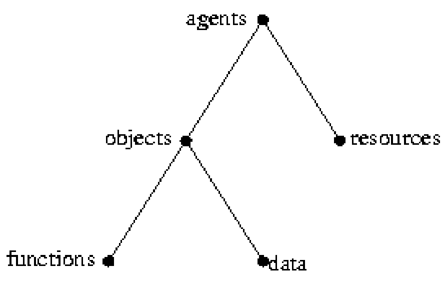

Fig. 6 Agent in Agent Oriented Paradigm

The following diagram depicts the categories of agents.

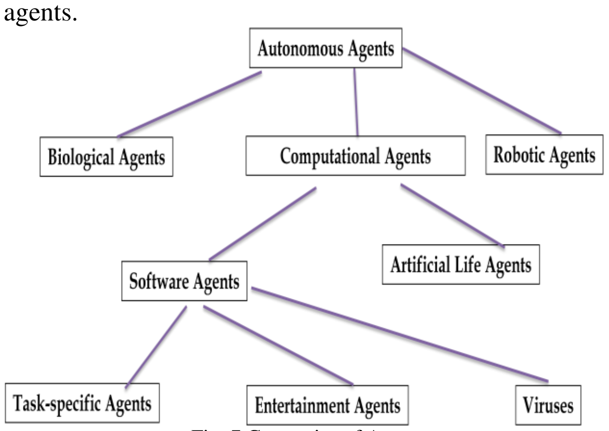

Fig. 7 Categories of Agents

V. WORKING MECHANISM OF INTELLIGENT AGENT

When an intelligent agent executes, it acquires input, process and produces output. Agent processing is domain-oriented. An agent "knows" about certain concepts, data structures, rules, and interfaces but is not necessarily capable of interpreting information outside its domain. An agent is capable of reasoning by referring encapsulated rules and transforms conditions into decisions. And it operates autonomously by good aspect of being persistent and capable of operating in a changing environment.

VI. AGENT ARCHITECTURE

The architecture of an agent should propose a methodology for building an autonomous agent. It should specify how the overall problem can be decomposed into interrelated sub-problems. It should specify how the construction of the agent can be decomposed into the construction of a set of component modules. It should specify how these component modules are made to interact. It specifies how these modules with interactions provide an answer to the question of how the sensor data and the current internal state of the agent determine the actions and the then internal state of the agent. Following figure shows the hybrid agent architecture.

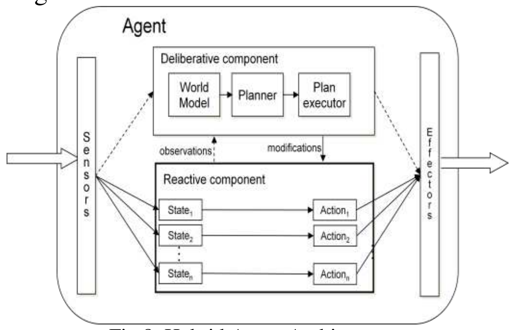

Fig.8 Hybrid Agent Architecture

VII. AGENT-OBJECT-RELATIONSHIP MODELING LANGUAGE (AORML)

In [3], an agent-oriented modeling language was proposed for the analysis and design of organizational information systems, called Agent-Object-Relationship modeling language (AORML). In the AORML, an entity can be an agent, an event, an action, a claim, a commitment, or an ordinary object. Special relationships between agents and events, actions, claims and commitments supplement the fundamental association, generalization and aggregation relationships of UML class models. AORML can be viewed as an extension of the Unified Modeling Language (UML). They author believes that AORML, by virtue of its agent-oriented categorization of different classes, allows more adequate models of organizations and organizational information systems than natural UML.

In [4], Gerd Wagner presents a UML profile for an agent-oriented modeling approach called an Agent-Object-Relationship (AOR). Casting the AOR metamodel as a UML profile allows AOR models to be notated using standard UML notation.

VIII. AGENT BASED DATABASE

From a modern perspective, a more general abstraction for a database is appropriate: a database is a computer model that is a source of faithful views of the external world. At any stage, the state of the computer model corresponds to the actual or conceived state of an external system. The user perceives the state of the computer model via views that rely upon suitable metaphors. The table metaphor that works well for a traditional recordkeeping system isn't a good way to describe a visual image. Modern computers have unexplored potential for new metaphors. Each metaphor involves different ways of presenting the state of the computer model to the user, and different ways of enabling the user to manipulate this state. In developing a computer model that represents a real-world state, the mode of presentation and manipulation is of the essence.

Modern database design requires very general modes of real-world simulation to support for intelligent decision making. This design to be effective the principles, concepts and procedures learnt from relational database theory alone cannot be suffice to frame the design in terms of tabular representations and relational operators. The ODB takes care of limitations of RDB. The limitations of ODB lie in the fact that the object does not take care of the prevailing changing environment around the object. This unattended aspect requires the empowerment of object to acquire the information from the dynamically changing object surrounding environment. The object having this capability becomes resourceful and we can call it as an agent.

IX. MODELLING ODB TO AODB

The discussed case of university database has set of objects like person, employee, faculty; staff, student, examination in-charge, timetable in-charge, stationary, leave, class room, course, teaching plan etc. Let us consider a scenario 1 and scenario 2.

A. *Scenario 1*

The examination in-charge as an object becomes resourceful if it acquires the information from objects in the surrounding environment such as stationary object regarding availability status of stationary, as staff object and its inherited class objects such as teaching staff, non-teaching staff, technical staff regarding availability status of staff, as class room object regarding availability of class room, as faculty and course objects regarding faculty wise course completion status, as student and examination form regarding the number of students appearing at the exam etc. The examination in-charge object becomes an agent for the database.

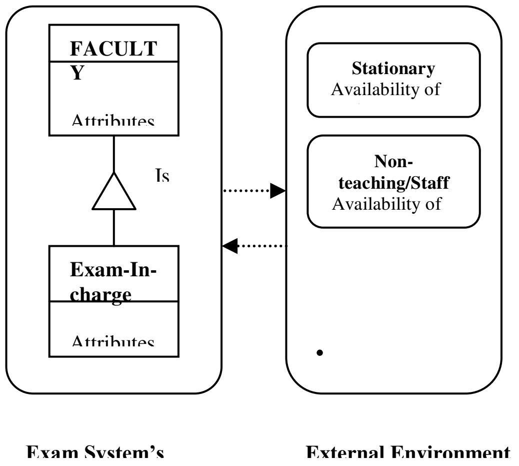

Fig 9 Agent Modeling in AODB which interact with object

B. *Scenario 2*,

If time-table in-charge wants to manage daily time-table, he requires information regarding daily class schedule, unavailability of faculty/(ies), timeslots to be manage, available faculties during that timeslot who has minimum load on that day.

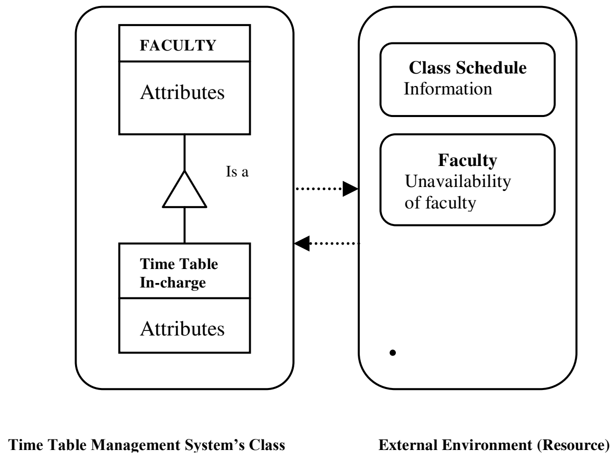
FACULTY
Attributes
Is a
Time Table In-charge
Attributes
Class Schedule
Information
Faculty
Unavailability of faculty
Time Table Management System's Class
External Environment (Resource)

Fig 10 Agent Modeling in AODB which interact with object

X. CONCLUSION

The transformation of RDB to ODB is analyzed with betterment in design to make the design nearer to the real life situations. The transformation of ODB to AODB takes an object to become more resourceful with required environmental data for decision making. Decision support system had utilized information from RDB as well as ODB for the purpose of decision making but AODB takes decision makers to a state further where agents as an autonomous agent does to the job of decision making and leads decision maker with information for effective intelligent decision making. The entire journey of effective decision making requires modeling of agent oriented database management system. This research can be extended in future to work on architecture and prototyping AODB for decision making systems, prototyping tracks and tool supports, analysis of tracks.

REFERENCES

[1]. M. Wooldridge. *Agents and software engineering*. In **AI*IA Notizie** XI(3), pages 31-37, September 1998.

[2]. Nicholas R. Jennings and Micheal J. Wooldridge, *Agent technology: foundations, applications, and markets* ,Pages: 3 - 28 ,Year of Publication: 1998

[3]. Wagner, G., *The Agent-Object-Relationship Metamodel: Towards a Unified View of State and Behavior*. May 2002, Technical Report, Eindhoven Univ. of Technology, Fac. of Technology Management. Available from http://AOR.rezearch.info.

[4]. Wagner, G., "A UML Profile for External AOR Models," in *A UML Profile for External AOR Models*, F. Giunchiglia, J. Odell, and G. Weiss, Eds. 2003, Springer-Verlag, Berlin. Available at http://www.auml.org/auml/working/Wagner-aose02-47.pdf.