The J. of MIS
Research
Vol.2, No.2, Dec 1992

객체지향 데이터베이스 설계를 위한 데이터 모델링에 관한 연구

박 원 옥*

A Study on Date Modeling for Object-Oriented Database Design

Object-Oriented method is a rather difficult one, theories are different as many as they are, and can hardly be found well-applied examples of OOD.

So, this paper deal with the history, the needs, the benefits, basic concept of object-oriented design methods, especially the designing procedure which covers from analyzing the requirements, designing the EER(extended entity relationship) model, transforming this model into relational schema, to OODB schema step.

Major contributions can be identified; First, the proposed method will ensure that it improve the productivity, increase the software quality, and elevate the maintainability. Second, it provide the framework of how object-oriented schema is constructed for OODB design. Third, suggested realistic case study will enhance the better understanding of how the current business process will be applied to the OODB design step.

Ⅰ. Introduction

The past decade has seen a rapid growth in database systems, not only with respect to the number of such systems, but also with regard to the amount of information being stored and the complexity of applications being developed. This growing demand for systems of ever-increasing complexity and precision has stimulated the need for higher level concepts,

* ㈜ 고려 전산실

tools and techniques for database design and development. Thus the database design methodologies which have been developed in recent years provided the designer with the means for modeling an enterprise at a high level of abstraction, before proceeding with the detailed logical and physical database design.

This study is aimed at the practicing database designer, the person who has to tackle real-world systems development projects every day. I assume that readers are concerned with the "middle" part of the development cycle : the activity of design. A designer may also be involved in the front-end activities of interviewing users to determine system requirements, as well as being involved in the back-end issues of system testing.

In this study, I am speaking primarily to the person responsible for designing overall database architecture by suggesting the object-oriented data modeling techniques.

1. Changes of Software Engineering Environments

During the past decade, there has occurred a lot of changes in the software engineering environments. Among many, I'll explain four major changes in the light of history of object -oriented approach. With this, you'll understand more easily why object-orientd concept has appeared.

1.1 Attention has gradually shifted from issues of coding to issues of design and analysis.

1.2 It was difficult to think about coding in object-oriented fashion when the language of choice was COBOL or plain-vanilla C; it has become easier with C++ and Smalltalk.

1.3 The systems built today are different from what they were ten or twenty years ago:they are larger, more complex, and more volatile. An object-oriented approach to analysis and design is likely to lead to a more stable system. Also, today's on-line, interactive systems devote much more attention to the user interface than the text-oriented batch processsing systems of the 1970s. An object-oriented approach to such systems—from analysis through design and into coding—is more natural way of dealing with such user-oriented systems.

1.4 The systems built today are more "domain-oriented" than the systems bulit in the 1970s and 1980s. Functional complexity is less of a concern than it was before : modeling the data has become a moderate priority; modeling problem domain understanding and systems responsibilities take higher priority.

2. The Needs and Objectives of Object-Oriented Design

2.1 There may be many advantages or objectives in case we adopt the object-oriented concept into organizations according to various theories. The first objectives of

OOD are to improve productivity; OOD focuses effort on the up-front activity of software design. In return for this investment, less time is required for testing and defect removal. But the overall productivity improvement during the development of a system may be a modest 20 percent, or even less if the project team is relatively unfamiliar with OOD.

But there is another perspective : instead of improving just the productivity of development, how about improving productivity across the entire life cycle! Most organizations recognize that 75 to 80 percent of a system's cost occurs after it has been deployed; this also means that many of the defects are discovered after deployment, and even more important, much of the functionality of the system is added after initial deployment.

Another is to increase software quality. Software quality will take on a new meaning in the 1990s, especially as the worldwide software industry becomes more competitive. As 'hard' defects in a software system fall below some threshold, end-users equate 'quality' with more than just the absence of defects. Software quality—fitness for use—begins to include ease of use, portability, and ease of modification. Thus, maintainability and the ability of a system to deal with continual change, will become more important aspects of software quality in the future.

Finally, the other may be to elevate maintainability. The requirements for a system will always be in a state of flux. Management may impose an artificial freezing of requirements at a particular point in time. But the true requirements, the needed system, will continue to evolve. Many forces affect this ever-changing requirements set : clients, competition, regulators, approvers, and technologists.

A designer endeavors to organize a design so that it is resilient to change; a packaging that will remain stable over time is sought.

2.2 Above all, the key objectives of OOD are improving productivity, increasing quality, and elevation maintainability; the organization may be able to reduce the number of people assigned to pure maintenance work, improve productivity or quality by providing a practical mechanism to reuse 'classes' from one project to another based upon a 'class library'. And also OOD make it easier for us to add, extend, change, delete or reuse some parts of systems whenever need them.

3. Motivations and Benefits of Object-Oriented

The motivations and benefits of object-oriented design are,

3.1 In the view of technical field;

We can tackle more challenging problem domains, it improves interactions between the problem-domain-expert and designer, it explicitly represent the commonality, it's resilient to change, and also we can reuse analyzed or designed results.

3.2 The other one we can consider is in the view of manager's side.

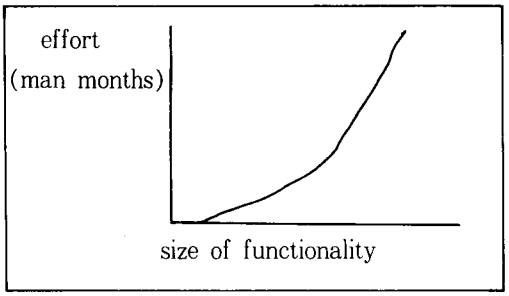


Figure 1.
(with conventional method)

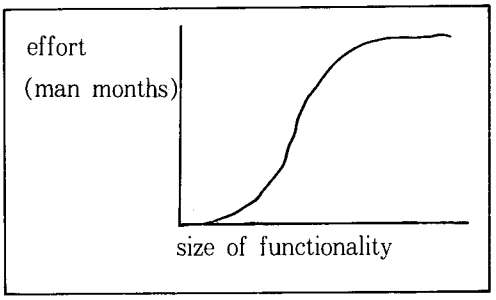


Figure 2.
(with object-oriented method)

Figure 1 illustrates the exponential time needed to develop a system(in terms of labor months) as a function of size or functionality. Using these object-oriented concepts in languages, databases, and use-interfaces, we will eventually be able to achieve a linear expansion in effort as a function of size or functionality[Setrag, 1990]. This is illustrated in figure 2 which shows that there is an initial rise in cost while learning the object-oriented technology and developing initial reusable components or class hierarchies, but later a more linear growth of effort is achieved [Setrag, 1990].

Ⅱ. Object-Oriented Date Modeling

1. Conventional vs Object-Oriented

Traditional systems development methodologies view an information system from two separate perspectives, the data perspective and the process perspective. The data perspective is concerned with entity-relationship modelling, relational analysis, and ultimately database schema generation and the physical implementation of the database. The process perspective is concerned with the functional requirements of the system and involves business activities, data flow diagrams, system functions and, at the low end, compilable units of program code(modules). In traditional tools for computer-aided software engineering(CASE tools) a data dictionary serves to integrate these two perspectives and maintain consistency(Figure 3)

In an object-oriented perspective(Figure 4), the principle of aggregation is centred on the underlying data abstractions. That is, every function must be associated with a particular object, and thus functions are grouped together if they operate on the same data abstraction. In this way, objects encapsulate both state and behaviour. Functions which are constituents of a higher level process may reside in different objects and a sequence of messages (or function calls) between objects is necessary to perform a higher level process.

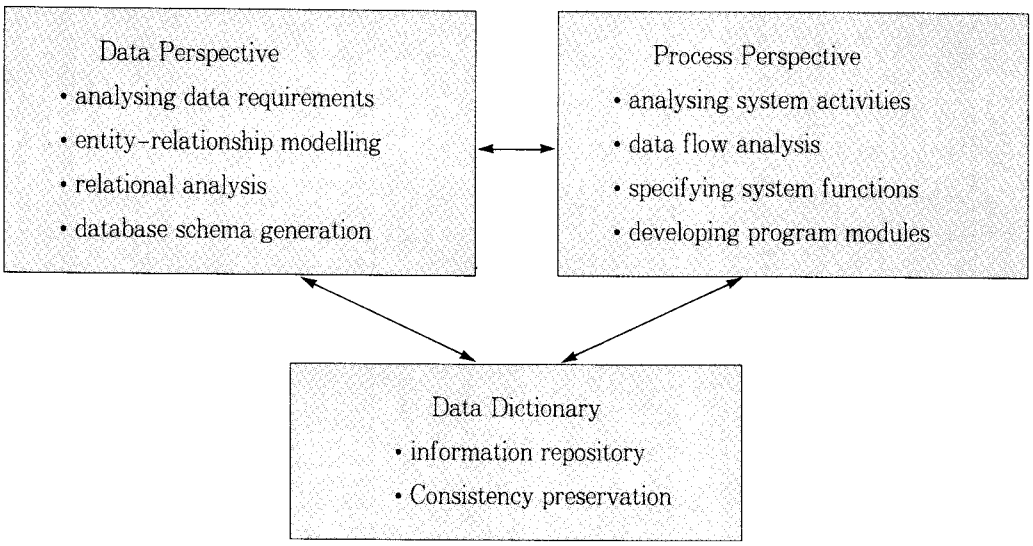


Figure 3. Conventional structured analysis

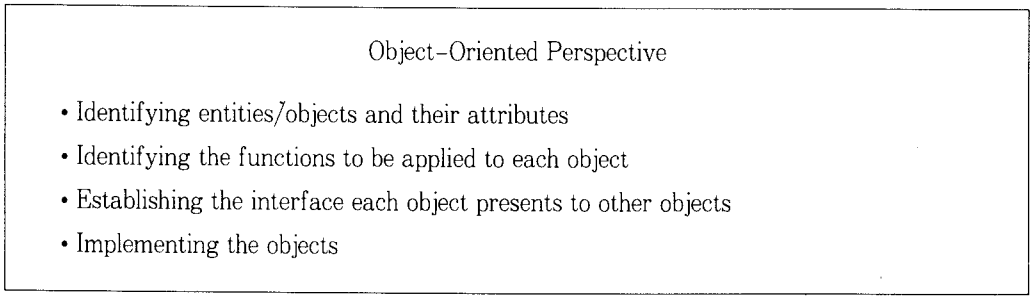


Figure 4. Object-oriented analysis

Because of the difference in aggregation principles, proceeding from a traditional structured analysis approach to an object-oriented design can be awkward. This can be avoided by adopting an object-oriented viewpoint during the analysis phase.

We find the object modeling techniques (OMT) is useful to model a system from three related but different viewpoints, each capturing important aspects of the system, but all required for a complete description.

The Object Modeling Technique(OMT) is our name for the methodology that combines these three views of modeling systems. The object model represents the static, structural, "data" aspects of a system. The dynamic model represents the temporal, behavioral, "control" aspects of a system. The functional model represents the transformational, "function" aspects of a system. A typical software procedure incorporates all three aspects : It uses data structures(object model), it sequ-

ences operations in time (dynamic model), and it transforms values(functional model). Each model contains references to entities in other models. For example, operations are attached to objects in the object model but more fully expanded in the functional model.

2. Object-Oriented System

There are some basics and principles of OOA that we should know before we enter into the object-oriented data modeling or database design. These are terminology, merging of disciplines, mapping, and the principles of managing complexity. The terminology "Object-Oriented" is difficult term, because the term "object" has been used in different ways within two very different disciplines :

- From Information Modeling, meaning a representation of some real-world thing, and some number of occurrences of that thing.
- From Object-Oriented Programming Languages, meaning a runtime instance of some processing and values, defined by a static description called a "class".

An equation for recognizing an object-oriented approach is :

Object-Oriented = Classes and Objects
+ Inheritance
+ Communication with messages

OOA builds upon the best concepts from information Modeling, Object-Oriented Programming Languages, and Knowledge-Based Systems – the concepts which have a solid basis in underlying principles for managing complexity.

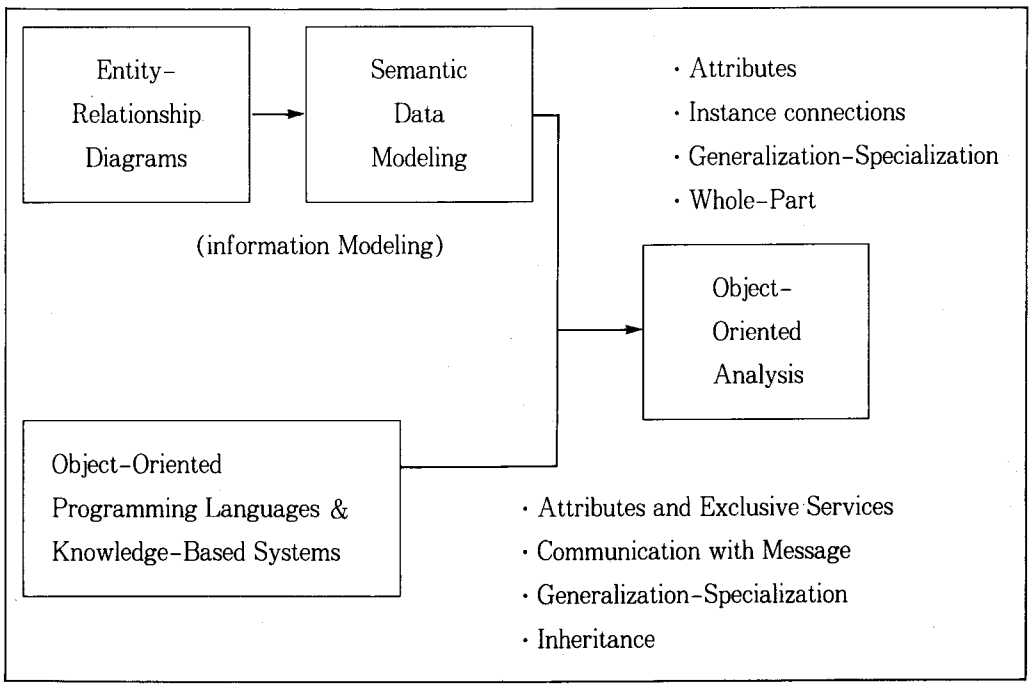


Figure 5. Merging of disciplines

From Information Modeling come constructs analogous to Attributes, Instance Connections, Generalization-Specialization, and Whole-Part. From Object-Oriented Programming Languages and Knowledge-Based Systems come the encapsulation of Attributes and exclusive Services, communication with messages, Generalization-Specialization and Inheritance.

OOA directly maps problem domain and system responsibility directly into a model. Instead of an indirect mapping from problem domain to flows and bubbles, the mapping is direct, from the problem domain to the model.

The last one is that OOA is not very helpful for systems with very limited responsibilities, or for systems with only one or two Class-& -Objects. For example, if a system takes an input stream, runs a ten page algorithm on it, and produces an output stream, then just a flow chart, a list of equations, and some terse text would be sufficient. The system's responsibilities are very limited.

Figure 6 shows that your understanding the concepts of the principles for managing complexity, compared with the analysis methods or OOA construct, will guide you easily to the object-oriented database schema designing step.

1 Abstraction
   a Procedural
   b Data
2 Encapsulation
3 Inheritance
4 Association
5 Communication with messages
6 Pervading methods of organization
   a Objects and attributes
   b Whole and parts
   c Classes and Members, and distinguishing between them
7 Scale
8 Categories of behavior
   a Immediate causation
   b Change over time
   c Similarity of functions

Figure 6. Principles for managing complexity
(with reference numbers, used in the tables which follow)

This table(figure 7) presents the differences between the various analysis methods, assessed in the light of principles for managing complexity :

Methods	Principles of Managing Complexity												
	1a	1b	2	3	4	5	6a	6b	6c	7	8a	8b	8c
Functional Decomposition	X												
Data Flow	X										X	X	
Information Modeling					X		X	X	X	X			
Object-Oriented	X	X	X	X	X	X	X	X	X	X	X	X	X

Figure 7. Analysis methods and the principles they incorporate

The purpose I quoted the relation between OOA methods-or-constructs and each principles of managing complexity is for your easier understanding OOD. Thirteen factors can be fully explained and applied in the light of object-oriented analysis. For example, in the information modeling method, abstraction (1a,1b),inheritance(3), and message communication(5) can not be applied yet, although these are most powerful concepts in OOD.

The following table(Figure 8) summarizes the OOA constructs which utilize these principles :

This table is not for explanation of what the OOA-construct contribute to principles of managing complexity, but just for your better understanding the concept and the relationship between them.

In an overall approach, OOA consists of five major activities.

OOA Construct	Principles of Managing Complexity												
	1a	1b	2	3	4	5	6a	6b	6c	7	8a	8b	8c
Class-&-Object		X	X				X						
Gen-Spec Structure				X					X				
Whole-Part Structure								X					
Attribute		X	X	X			X		X				
Service	X	X	X	X					X		X	X	X
Instance Connection					X								
Message Connection			X			X					X		
Subject								X		X			

Figure 8. OOA's application of principles for managing complexity

Finding Class-&-Object
Identifying Structures
Identifying Subjects
Defining Attributes
Defining Services

These are indeed activities, not sequential steps. The activities guide the analyst from high levels of abstraction(e.g., problem-domain Class-&-Objects) to increasingly lower levels of abstraction(Structures, Attributes, and Services). And the ordering of these five activities represents the most common overall approach. But, In this study, let alone the approach to OOA methodology, I'll directly touch the Object-Oriented data modeling with case study "Hospital Database".

3. Case Study

Now, in this section, I'll show you how to design both relational and object-oriented database schema. With the hospital business systems, analyzing the requirements, designing the EER(extended entity relationship) model, transforming this model into relational schema, and finally we can obtain the object-oriented database schema.

3.1 Requirements analysis

A requirements analysis yields the following informal description of the information to be recorded

The patients occupying each ward

Most patients are assigned to a ward on admittance and each ward may contain many patients. However, consultants(senior surgeons)at the hospital may have private patients who are assigned to private rooms, each of which has a unique identification number. The information to be recorded about a patient upon registration at the hospital includes a unique patient number, name, address, sex, phone number, date of birth and blood group.

The nurses assigned to each ward

A nurse may or may not be assigned to a ward and he/she cannot be assigned to more than one ward. A ward may have many nurses assigned to it. Nurses are identified by their staff numbers and their names, addresses, phone numbers and grades are also recorded. Each ward has a unique number and is dedicated to a particular type of patient(e.g. geriatric, pediatric, maternity, etc.).

The operations undergone by patients

A patient may have a number of operations. The information to be recorded about an operation includes the type of operation, the patient, the surgeons involved, date, time and location.

The surgeons who perform operations

Only one surgeon may perform an operation, any other surgeons present being considered as assisting at the operation. Surgeons come under the direction of senior surgeons, called consultants, who may also perform or assist at operations. Information recorded

about a surgeon includes name, address and phone number. Each consultant has a specialism.

The theatres in which operations are performed

An operation can be performed in only one theatre but a given theatre may be the location of many operations. Each theatre has an identifying number and some may be specially equipped for certain classes of operation.

The nurses assigned to each theatre

A nurse may or may not be assigned to a theatre and he/she cannot be assigned to more than one theatre. A theatre may have many nurses assigned to it.

3.2 The entity-relationship model

The following is a list entities, attributes and relationships which represent the informal description of the database outlined above [Tim & John, 1990];

- Entity type SURGEON, with attributes NAME, ADDRESS and PHONE-NO.
- Entity type CONSULTANT which is a subtype of the entity type SURGEON. Every consultant is a specialist in a particular branch of surgery and this is recorded as an additional attribute SPECIALITY.
- Entity type PATIENT, with attributes PATIENT#(a unique patient number), NAME, ADDRESS, PHONE-NO, DATE-OF-BIRTH, SEX and BLOOD-GROUP.
- Entity type PRIVATE-PATIENT which is a subtype of the entity type PATIENT. The number of the private room to which such a patient is assigned is recorded as an additional attribute, ROOM#.
- Entity type NURSE, with attributes STAFF #(a unique staff number), NAME, ADDRESS, PHONE-NO, SEX and GRADE. A nurse may be assigned either to a ward a theatre.
- Entity type WARD, with attributes WARD #(a unique ward number), WARD-TYPE and NO-OF-BEDS.
- Entity type THEATRE, with attributes WARD#(a unique theatre number) and THEATRE-TYPE.
- Entity type THEATRE, with attributes OPERATION-TYPE, DATE and TIME.

Appropriate relationships, as extracted from the requirements analysis, are as given in the following list :

PERFORMS, a 1 : N relationship between entity types SURGEON and OPERATION, with the entity type OPERATION being a mandatory member of this relationship, i.e. every operation must be performed by a surgeon.

ASSISTS, an N : M relationship between entity types SURGEON and OPERATION indicating those surgeons who assist at each operation. A possible attribute of this relationship is the ROLE played by each surgeon.

SUPERVISES, a 1 : N relationship between the entity types CONSULTANT and SURGEON. The membership class of SUR-

GEON in this relationship is optional since there may be surgeons(e.g.consultants) that are not supervised.

TREATS, a 1 : N relationship between the entity type CONSULTANT and the subtype PRIVATE-PATIENT. The membership class of PRIVATE-PATIENT in this relationship is mandatory. That is, every private patient is assigned to a consultant for treatment.

UNDERGOES, a 1 : N relationship between entity types PATIENT and OPERATION. The entity type OPERATION is a mandatory member of this relationship since an operation must always have an associated patient.

OCCUPIES, a 1 : N relationship between entity types WARD and PATIENT, where the membership class of PATIENT is 'almost' mandatory since most patients are assigned to a ward on entry to the hospital.

LOCATED, a 1 : N relationship between entity types THEATRE and OPERATION. The entity type OPERATION is a mandatory member of this relationship since clearly every operation must be located in a theatre.

WARD-ASSIGN, a 1 : N relationship between entity types WARD and NURSE. A possible attribute for this relationship is DATE-ASSIGNED giving the date on which a particular nurse was assigned to a ward. The NURSE entity type is an optional member of this relationship since a nurse may or may not be assigned to a ward at any given time.

THEATRE-ASSIGN, a 1 : N optional relationship between THEATRE and NURSE, with attribute DATE-ASSIGNED giving the date on which a particular nurse was assigned to a theatre. As is the case with WARD-ASSIGN, the NURSE entity type is an optional member of this relationship since a nurse may or may not be assigned to a theatre at any given time.

In addition to the above relationships, we have an IS_A relationship between the subtype CONSULTANT and the entity type SURGEON, and another between the subtype PRIVATE-PATIENT and the entity type PATIENT. A schematic EER model for the hospital application is illustrated in Figure 9 [Fim & John, 1990].

3.3 The relational schema

After applying the guidelines(Converting an EER model into a relational schema), we abtained the following normalized relatioal schema.

In the relational schema the relationship OCCUPIES is represented by the foreign key WARD# in the PATIENT relation since this relationship is "almost mandatory" for PATIENT. That is, most patients are assigned to wards. The relationship TREATS is represented by the foreign key SNAME(the name of the consultant)in the PRIVATE-

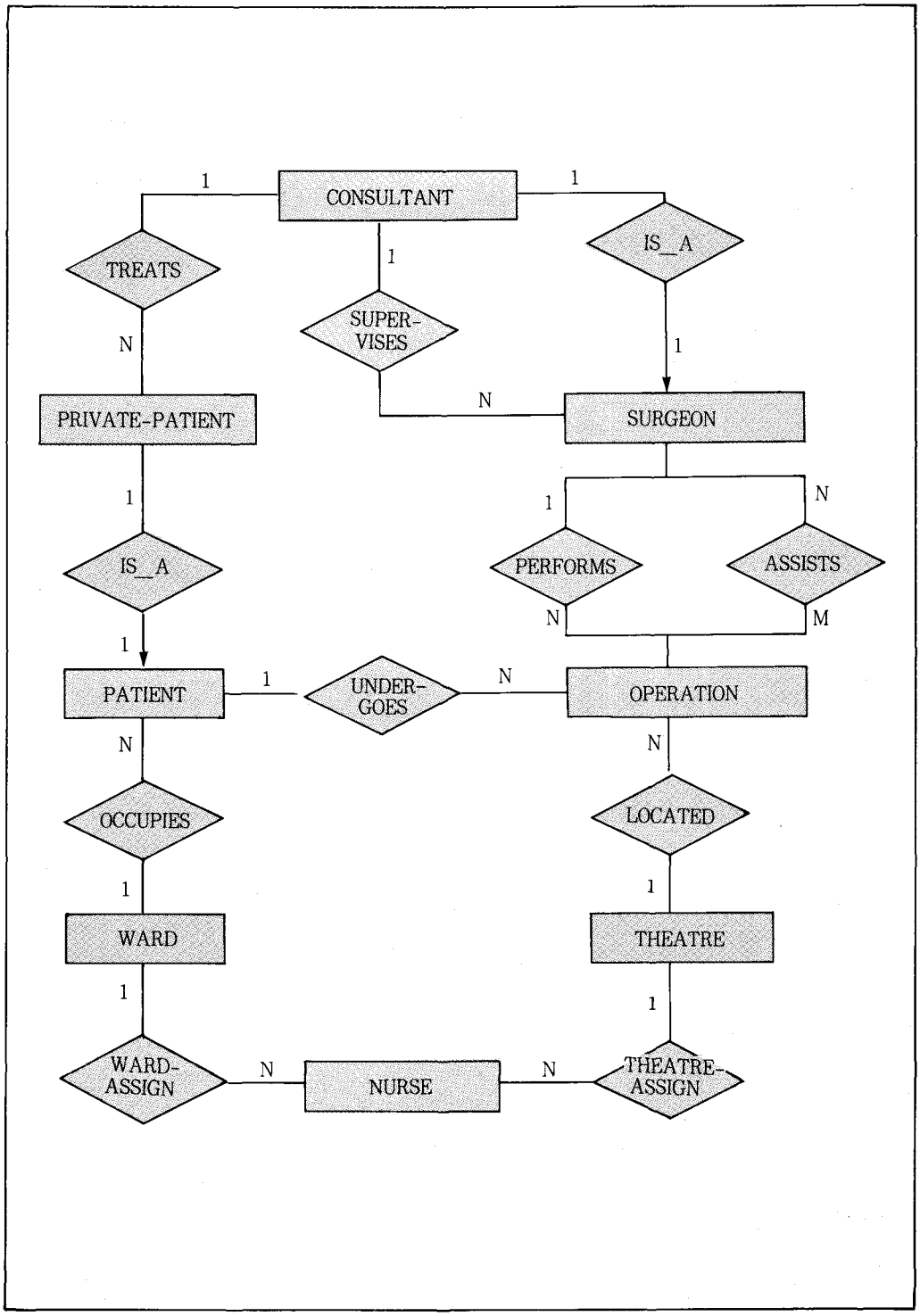


Figure 9. EER model for the hospital database

RELATIOAL SCHEMA

SURGEON (SNAME, ADDRESS, PHONE-NO)
CONSULTANT(SNAME, SPECIALITY)
PATIENT(PATIENT#, WARD#, PNAME, ADDRESS, PHONE-NO,
DATE-OF-BIRTH, SEX, BLOOD-GROUP)
PRIVATE-PATIENT(PATIENT#, SNAME, ROOM#)
NURSE(STAFF#, NNAME, ADDRESS, PHONE-NO, SEX GRADE)
THEATRE(THEATRE#, THEATRE-TYPE)
OPERATION(OP#, SNAME, THEATRE#, PATIENT#, OP-TYPE, DATE, TIME)
SUPERVISES(SURGEON-SNAME, CONSULTANT-SNAME)
ASSISTS(OP#, SNAME, ROLE)
WARD-ASSIGN(STAFF#, WARD#, DATE-ASSIGNED)
THEATRE-ASSIGN(STAFF#, THEATRE#, DATE-ASSIGNED)

PATIENT relation. Also, the foreign keys SNAME, PATIENT# and THEATRE# in the OPERATION relation represent respectively the PERFORMS, UNDERGOES and LOCATED relationships of which OPERATION is a mandatory member.

The M : N relationship ASSISTS is represented by a separate relation scheme containing the key attributes of SURGEON and OPERATION, together with the attribute ROLE which indicates the role played by a surgeon at an operation.

The optional 1 : N relationships WARD-ASSIGN and THEATRE-ASSIGN are each represented by a separate relation scheme containing the key attributes of the participating entity sets together with the additional attribute DATE-ASSIGNED. If the majority of nurses were assigned to wards we might choose to represent the WARD-ASSIGN relationship by posting the foreign key WARD# into the NURSES relation (together with the attribute DATE-ASSIGNED).

The optional 1 : N relationship SUPERVISES is represented by a separate relation, but could be represented by posting the name of the supervising consultant(C-SNAME) into the SURGEON relation, as follows :

SURGEON(SNAME, SADDRESS, PHONE-NO, C-SNAME)

The attribute C-SNAME in this case will be null for all those surgeons who do not come under the direction of a consultant.

3.4 The object-oriented schema

The object-oriented database schema is a

set of class definitions, one for each entity in the EER model. Relationships are typically represented by properties(or operations) rather than separate classes, unless the relationship itself has attributes(apart from those of the participating entities). In this case a class may be constructed for the relationship, For the hospital database the many-to-many relationship ASSISTS between SURGEON and OPERATION is probably best incorporated in the class for surgeon.

Once again it is important to note that all the information associated with a particular entity is represented in the class definition, rather than being scattered across several relations. For example, in the class Surgeon we have represented not only the simple properties of surgeons(name, address, phone number)but also the supervising consultant and the operations the surgeon has performed and assisted at.

An abstract data type(ADT) defines a class of objects as follows with which you may understand how the object-oriented database schema is constructed.

```
class Surgeon
  properties
      name, address, phone_no : String
      sex : (Male, Female);
      supervised_by : Consultant
          inverse is Consultant.supervises;
      performs : Set(Operation)
          inverse is
          Operation.performed_by;
      assists_at : Set(Operation)
          inverse is Operation.assisted_by;
  operations
      create(...);
      assign_duties(...);
      role(Surgeon, Operation)→ RoleType;
end Surgeon.

class Consultant
  inherit Surgeon
  properties
      supervise : Set(Surgeon)
          inverse is Surgeon.supervised_by
      treats : Set(Private_Patient)
              inverse is
              Private_Patient.treated_by;
  operations
      create(...);
      calculate_fees(hours,...);
end Consultant.

class Patient
  properties
      number : Interger;
      name, address, phone_no : String;
      sex : (Male, Female);
      date_of_birth : Date;
      blood_group : Blood_Type;
      on_ward : Ward
          inverse is Ward.patients;
      undergoes : Set(Operation)
          inverse is
          Operation.performed-on;
  operations
      create(...);
      admit(...);
```

```
        discharge(...);
        ...
end Patient.

class Patiente_Patient
  inherit Patient
  properties
        room# : Integer;
        insurance : InsuranceType;
        treated_by : Consultant
                        inverse is
                        Consultant.treats;
  operations
        create(...);
        calculate_charges(...);
        ...
end Private_Patient.

class Ward
  properties
        ward# : Integer;
        no_of_beds : Integer;
        occupancy : Integer;
        type : (Geriatric, Pediatric,
        Maternity,...);
        patients : Set(Patient)
                    inverse is Patient.on_ward;
        nurses : Set(Nurse)
                    inverse is Nurse.ward-assign;
  operations
        create(...);
end Ward.

class Operation
  properties
        date : date;
        type : OperationType;
        performed_on : Patient
                inverse is Patient.operations;
        performed_by : Surgeon
                inverse is Surgeon.performs;
        assisted_by : Set(Surgeon)
                inverse is Surgeon.assists_at;
        located_in : Theatre
                inverse is Theatre.holds;
  operations
        create(...);
        schedule(...);
        cancel(...);
        ...
end Operation.

class Nurse
  properties
        staff#, name, address, phone# :
        String;
        sex : (Male, Female);
        grade : (Student, SEN, SRN,...);
        ward_assign : Ward
                inverse is Ward.nurses;
        theatre_assign : Theatre
                inverse is Theatre.nurses;
  operations
        create(...);
        ...
end Nurse.

class Theatre
  properties
        theatre# : Integer;
        type : TheatreType;
        nurses : Set(Nurse)
```

```
        inverse is Nurse.theatre_assign;
    holds : Set(Operation)
        inverse is Operation.located_in;

    operations
        create(...);
        ...

end Theatre
```

Although it is possible to implement ADTs in a language that provides no special support for them, it is more convenient if an abstract type can be implemented as a single program module. In object-oriented languages an ADT is implemented via such a module, which is usually called a class. For example, in C++ a class definition takes the following form :

```
class name
{
private :
    private components
public :
    public components
};
```

Ⅲ. Conclusion

Object-oriented system design is primarily based upon data abstraction(class-&-object), inheritance, and message communication.

In this study, I showed the relationship between data abstraction(entity) and object-oriented database design. I also showed that the extended entity relationship(EER) approach, which include the notions of generalization and subtyping, is a valuable first step towards an object-oriented data model.

First, In the object-oriented perspective every operation on the database must be associated with a particular object. Thus functions are grouped together if they operate on the same data abstraction. In this way, objects encapsulate both state and behaviour (Data type).

Second, The relational model is incapable of express integrity constraints with greater semantic content than straightforward referential integrity. For example, it is not possible to express the fact that a relationship is one-to-one or one-to-many.

Such constraints must be built into the application code that manipulates the relational database. Since such code is not generally shared among all applications, it is difficult to ensure that data will be updated consistently at all times.

By contrast, in the object-oriented model a class defines a data abstraction and this abstraction includes a specification of the operations(methods) that can be applied to instances of the class. By defining the database in terms of such abstrations a high degree of data independence is achieved. That is, it is possible to alter the way in which a class is implemented without affecting other classes or transactions that make use of the abstraction.

Entity integrity is also handled rather

differently in object-oriented systems. All objects(class instantiations)have a unique identity and other objects can refer to that identity(Data integrity).

Third, Relational database management systems tend to offer very limited facilities for the expansion or modification of existing data structures. But, The tight coupling between appications and data in the object-oriented model offers considerably more scope for schema evolution through the extension and refinement of existing data structures and the effective reuse of applications code(Schema evolution).

Fourth, Data manipulation in an object-oriented database system is accomplished by means of the operations defined in the class interfaces and through the constructs provided by the programming language surrounding the class definitions. However, many object-oriented systems also provide high level query language interfaces, mostly based on SQL(Data manipulation).

The relative merits of the relational and object-oriented approaches to data mangement may be summarized into four as above (data types, data integrity, schema evolution and data manipulation). These are main contribution factors to improve productivity, software quality, to decrease maintenance cost in Object-Oriented.

REFERENCES

Fred R. Mcfadden & Jeffrey A. Hoffer, *Database management*, The Benjamin Cummings Publishing Co.

James Rumbaugh, M. Blaha, W. Premer-lani, F. Eddy, W. Lorensen, *Object-Oriented Modeling and Design*. Prentice-Hall International, 1991, pp.15-18.

John G Hughes(The university of Ulstar), *Object-Oriented Database*, Prentice hall international.

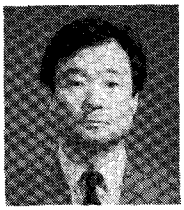
Peter Coad & Edward Yourdon, *Object-Oriented Analysis 2nd Editon*. Prentice-hall International, Inc.

Peter Coad & Edward Yourdon, *Object-Oriented Design*. Prentice-Hall international, Inc. pp.14-17.

Setrag Khoshafian & Razmik Abnous, *Object Orientation*. Wiley, 1990, pp.2-5.

Tim Korson & John D. Mcgregor, "Understanding Object-Oriented : A Unifying paradigm", *Communications of the ACM*, September, 1990, VOL.33, No.9. pp.42-50.

◇ 저자소개 ◇



저자 박원옥은 고려대학교 경영대학원에서 경영정보를 전공했으며, 근무처는 대우 그룹의 비계열사인 ㈜고려이며, 전산실 책임자로 재직중이다. 그는 럭키개발㈜, 대우자동차㈜ 전산실에서 프로그래머, 시스템 분석가 과정을 두루 거치면서 Network DB, Hierarchical DB, RDB 등을 모두 사용해 오던 중 OODB에 관심을 갖게 되었으며, 학위논문으로는 "실업무 중심의 객체지향분석에 관한 연구(A Study on Object-Oriented Aualysis)"이다. 그의 관심 분야는 객체지향 분석, 설계이다.