Design and Implementation of a Relational Database for Management of UML Models
Abstract

1. Introduction

2. UML
   2.1 UML
   2.2 UML (Class Diagram)
   2.3 UML

3. UML
   3.1 UML
   3.2 UML
   3.3 UML

4. UML
   4.1 UML
   4.2 UML
   4.3 UML
   4.4 UML

5. Conclusion

- i -
Design and Implementation of a Relational Database for Management of UML Models

Seong Dae Lee

Department of Computer Engineering, Korea Maritime University, Pusan, Korea

Abstract

The UML (Unified Modeling Language) is a modeling language for specifying, visualizing, constructing, and documenting systems. It also supports systematically the design and development of systems. There may be a large number of models to be managed in a real modeling environment. A methodology to save and retrieve the models effectively, therefore, needs to be developed.

This thesis focuses on the class diagram which is the core part of UML. It proposes a database supported methodology for the management of class diagrams. In the proposed methodology, class diagrams are saved in and retrieved from a relational database. To save a class diagram in the database, the constitution of the class diagram is translated in terms of relational tables. To retrieve a class diagram from the database, the user-specified query is translated into the SQL (Structured Query Language), and then the constituent of the class diagram is searched from the tables in the
database.

The proposed methodology can exploit the function of the relational database such as managing a large number of models, sharing the models among users, and fast queries. It, therefore, provides a powerful framework for an effective management of UML models and a fast development of systems.
1. UML (Unified Modeling Language) framework guideline

UML [Rumbaugh, OMT [1], Booch, Booch [2,3], Jacobson, OOSE [4]] has been released since 1989 with OMG (Object Management Group). UML 1.1 was released in 1997, and UML 2.0 RFI [5] was released in 1999. UML 2.0 RF is included in these releases.
UML类图是软件工程中的一种重要工具，用于描述系统的静态结构。Rational Rose是一个流行的UML建模工具。Plastic是一个基于XML的UML模型存储器。

UML类图展示了类（class）、属性（attribute）、操作（operation）、关系（relationship）、关系模式（relational schema）、主键（primary key）、外键（foreign key）等概念。这些概念在模型中表示了系统的静态结构。

2. UML类图可以表示系统的静态结构。3. UML类图是软件工程中的一种重要工具。
UML は以下のようにして使用されます。4
UML は以下のようにして使用されます。5
UML は以下のようにして使用されます。5
2. UML

UML (visualizing), (specifying), (constructing) (documenting) (visualizing), (specifying), (constructing) (documenting)

(visualizing), (specifying), (constructing) (documenting)

(visualizing), (specifying), (constructing) (documenting)

2.1 UML

UML (visualizing), (specifying), (constructing) (documenting) 3 (visualizing), (specifying), (constructing) (documenting) (Structural Things), (Behavioral Things), (Grouping Things), (Annotation Things)

(visualizing), (specifying), (constructing) (documenting) (Structural Things), (Behavioral Things), (Grouping Things), (Annotation Things)

(visualizing), (specifying), (constructing) (documenting) (Structural Things), (Behavioral Things), (Grouping Things), (Annotation Things)

(visualizing), (specifying), (constructing) (documenting) (Structural Things), (Behavioral Things), (Grouping Things), (Annotation Things)

(visualizing), (specifying), (constructing) (documenting) (Structural Things), (Behavioral Things), (Grouping Things), (Annotation Things)

(visualizing), (specifying), (constructing) (documenting) (Structural Things), (Behavioral Things), (Grouping Things), (Annotation Things)

(visualizing), (specifying), (constructing) (documenting) (Structural Things), (Behavioral Things), (Grouping Things), (Annotation Things)

(visualizing), (specifying), (constructing) (documenting) (Structural Things), (Behavioral Things), (Grouping Things), (Annotation Things)

(visualizing), (specifying), (constructing) (documenting) (Structural Things), (Behavioral Things), (Grouping Things), (Annotation Things)

(visualizing), (specifying), (constructing) (documenting) (Structural Things), (Behavioral Things), (Grouping Things), (Annotation Things)

(visualizing), (specifying), (constructing) (documenting) (Structural Things), (Behavioral Things), (Grouping Things), (Annotation Things)

(visualizing), (specifying), (constructing) (documenting) (Structural Things), (Behavioral Things), (Grouping Things), (Annotation Things)

(visualizing), (specifying), (constructing) (documenting) (Structural Things), (Behavioral Things), (Grouping Things), (Annotation Things)

(visualizing), (specifying), (constructing) (documenting) (Structural Things), (Behavioral Things), (Grouping Things), (Annotation Things)

(visualizing), (specifying), (constructing) (documenting) (Structural Things), (Behavioral Things), (Grouping Things), (Annotation Things)

(visualizing), (specifying), (constructing) (documenting) (Structural Things), (Behavioral Things), (Grouping Things), (Annotation Things)

(visualizing), (specifying), (constructing) (documenting) (Structural Things), (Behavioral Things), (Grouping Things), (Annotation Things)

(visualizing), (specifying), (constructing) (documenting) (Structural Things), (Behavioral Things), (Grouping Things), (Annotation Things)

(visualizing), (specifying), (constructing) (documenting) (Structural Things), (Behavioral Things), (Grouping Things), (Annotation Things)

(visualizing), (specifying), (constructing) (documenting) (Structural Things), (Behavioral Things), (Grouping Things), (Annotation Things)

(visualizing), (specifying), (constructing) (documenting) (Structural Things), (Behavioral Things), (Grouping Things), (Annotation Things)

(visualizing), (specifying), (constructing) (documenting) (Structural Things), (Behavioral Things), (Grouping Things), (Annotation Things)

(visualizing), (specifying), (constructing) (documenting) (Structural Things), (Behavioral Things), (Grouping Things), (Annotation Things)
(Collaboration), (Statechart), (Activity) (Implementation) (Static Structure) (Class), (Object) (Implementation) (Component), (Deployment) [10,11,12,13].

Fig. 2.1 Building Blocks of UML
UML(機制)

UML(制限)

UML(型)

UML(テキストストリング)

UML(付加値)

2.2 UML (クラスダイアグラム)
2.2.1 (Class) (Interface)

UML provides the capability to define classes and interfaces [5,6,10]. UML 2.2 specifies the class and interface (name), (attribute list), (operation list), (visibility) of classes and interfaces. The visibility of the attributes (visibility) can be either public, private, or protected.

(transaction) (operation) (attribute) (visibility) (stereotype)

Fig. 2.2 Representation of UML Class
2.2.2 （Relationship）

(dependency), (generalization), (association), (aggregation) (realization) (operation) “is-a” “kind-of”
2.4 Relationship of Class Diagram

2.4.1 “Window”가 “Event”로 “has-a” 관계를 나타내고 있습니다. 
“ConsoleWindow”가 “DialogBox”로 “Window”로 “part-of” 관계를 나타내고 있습니다. 
“Control”가 “DialogBox”로 “Window”로 “part-of” 관계를 나타내고 있습니다.

Fig. 2.4 Relationship of Class Diagram
2.3 UML

The Unified Modeling Language (UML) is a standard OMG standard for object-oriented modeling. It was introduced in 1990, with the first edition of the specification published in 1995. The latest version, UML 2.0, was released in 2005.

UML was developed by a group of object-oriented modeling experts, including Grady Booch, Ivar Jacobson, and James Rumbaugh. The first version of the language was called Booch94, after its main author. UML has since evolved through several revisions, with the latest being UML 2.5, which was released in 2013.

Booch, Rumbaugh, Jacobson, and their colleagues have written several books and papers on UML, including "Object-Oriented Analysis and Design with Applications" (Booch, 1994), which introduced the Booch94 notation, and "Object-Oriented Modeling and Design" (Rumbaugh, Jacobson, 1998), which introduced the Rumbaugh90 notation.

UML is widely used in software engineering, particularly in the development of large-scale software projects. It provides a standard notation for modeling software systems, which can be used by developers, analysts, and stakeholders to communicate about the system's design and behavior.
Rose[6] is "Rational Rose Enterprise Edition 2000". Rose is used in the development of C++, Java, and other languages. UML is a notation for object-oriented software development (forward engineering) and is used in the development of object-oriented software (reverse engineering) [6].

Plastic[7] is a UML tool used in the development of Java (Object) and other languages. The development of Java is based on 8 [7].
3 UML

UML  is  a  widely  used  notation  for  modeling  software  systems.  It  consists  of  several  parts:  attributes  (attribute)  and  operations  (operation).  Attributes  and  operations  are  represented  in  different  ways  depending  on  the  context  of  the  model.  Attributes  are  used  to  describe  the  properties  of  objects,  while  operations  are  used  to  describe  the  behaviors  of  objects.

3.1 UML

The  notation  D = \{d_1, d_2, ..., d_k\} represents  a  set  of  attributes,  where  \(d_x\)  is  an  attribute  ID  (\(x = 1, 2, ..., k\)).
C = \{c_1, c_2, \ldots, c_l\}

\begin{align*}
C & = \{\text{class name}, d_i\}, \ldots, d_i, c_i, \ldots, c_l, \ldots
\end{align*}

\begin{align*}
D & \subseteq C (x, x = 1, 2, \ldots, l).
\end{align*}

\begin{align*}
A & = \{a_1, a_2, \ldots, a_m\}, A \subseteq A (a_t, x = 1, 2, \ldots, m).
\end{align*}

\begin{align*}
O & = \{o_1, o_2, \ldots, o_n\}, O \subseteq O (c_i, x = 1, 2, \ldots, n).
\end{align*}

\begin{align*}
\text{is interface} & \begin{cases} \text{TRUE}, \text{FALSE} \end{cases}
\end{align*}
ÀÎÅÍÆäÀ̽º´Â±×¸²

2.3°ú°°À̵ΰ¡ÁöÇüÅ·ÎÇ¥ÇöµÈ´Ù

µû¶ó¼­´Ù¸¥½ºÅ×·¹

È®Àå¿¡ÇÊ¿äÇѺκеé

Àº½ÇÁ¦±¸Çö¿¡¼­°í·ÁÇÏ¿´´Ù

°¢Å¬·¡½ºÀÇ»ó¼Ó¼º°ú¿¬°ü¼ºÀ»¼³Á¤Çϴ°ü°èÀÇÁýÇÕÀº´ÙÀ½°ú°°ÀÌÁ¤ÀÇÇ önünde ÀÇÇชิ้น.

È®Àå¿¡ÇÊ¿äÇѺκÐÀ¸·Î´Â±ÔÄ¢

ÀÎÀÚ(relationship), ÌÎŬ·¡½º°¡ÀÖÀ½À»³ªÅ¸³½´Ù.

[^6] R = \{r_1, r_2, ..., r_o\} (return type), \( r_x = (c_i, c_j, relationship, d_i) \) (parameter)

ÀÇÀÓÀÇÀÇ¿ø¼ÒÀΰü°è\( r_x \) = (aggregation), \( c_i \) (super class), \( c_j \) (sub class)

3.1“Company” UML\( \) Department, Office\( \) (multiplicity) "1", Department, Office\( \) "1..*". Company
3.1 Company, Department, Office, Person, Department, Person

"member" "manager" "subset" "manager", "member"

3.1 An Example of a Class Diagram: Company

Fig. 3.1 An Example of a Class Diagram: Company
１) 3.1.1 3.1.2 3.1.3 3.1.4 3.1.5 3.1.6 3.1.7 3.1.8

5

D = \{Company\}

C = \{(Company, False, Company),
       (Department, False, Company),
       (Office, False, Company),
       (Person, False, Company),
       (ContactInformation, False, Company),
       (PersonnelRecord, False, Company),
       (Headquarters, False, Company),
       (ISecureInformation, True, Company)\}

(attribute)

A = \{(name, Department),
       (address, Office),
       (voice, Office),
       (name, Person),
       (employeeID, Person),
       (title, Person),
       (address, ContactInformation),
       (taxID, PersonnelRecord),
       (employmentHistory, PersonnelRecord),
       (salary, PersonnelRecord)\}
\[ O = \{(\text{getPhoto}, \text{Person}), \]
\[ (\text{getSoundBite}, \text{Person}), \]
\[ (\text{getContactInformation}, \text{Person}), \]
\[ (\text{getPersonnelRecord}, \text{Person})\} \]

\[ R = \{(\text{Company}, \text{Department}, \text{aggregation}, \text{Company}), \]
\[ (\text{Company}, \text{Office}, \text{aggregation}, \text{Company}), \]
\[ (\text{Department}, \text{Department}, \text{aggregation}, \text{Company}), \]
\[ (\text{Department}, \text{Office}, \text{association}, \text{Company}), \]
\[ (\text{Department}, \text{Person}, \text{association}, \text{Company}), \]
\[ (\text{Department}, \text{Person}, \text{association}, \text{Company}), \]
\[ (\text{Office}, \text{Headquarters}, \text{generalization}, \text{Company}), \]
\[ (\text{Person}, \text{ContactInformation}, \text{dependency}, \text{Company}), \]
\[ (\text{Person}, \text{PersonnelRecord}, \text{dependency}, \text{Company}), \]
\[ (\text{PersonnelRecord}, \text{ISecureInformation}, \text{association}, \text{Company})\} \]

\[ 3.2 \quad D, \ A, \ O, \ R \quad \text{(relational model)} \quad \text{relational schema} \]
$D$-schema = (diagram_name)

$C$-schema = (class_name, is_interface, diagram_name)

$A$-schema = (attribute_name, class_name)

$O$-schema = (operation_name, class_name)

$R$-schema = (super_class, sub_class, relationship, diagram_name)

[8] $\text{\quad (tuple)}$

3.2 $\text{\quad Company}$ 3.1 $\text{\quad Company}$
### Converted Database Tables

#### D (Diagram) Table

<table>
<thead>
<tr>
<th>diagram_name</th>
<th>Company</th>
</tr>
</thead>
</table>

#### C (Class) Table

<table>
<thead>
<tr>
<th>class_name</th>
<th>is_interface</th>
<th>diagram_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company</td>
<td>False</td>
<td>Company</td>
</tr>
<tr>
<td>Department</td>
<td>False</td>
<td>Company</td>
</tr>
<tr>
<td>Office</td>
<td>False</td>
<td>Company</td>
</tr>
<tr>
<td>Person</td>
<td>False</td>
<td>Company</td>
</tr>
<tr>
<td>ContactInformation</td>
<td>False</td>
<td>Company</td>
</tr>
<tr>
<td>PersonnelRecord</td>
<td>False</td>
<td>Company</td>
</tr>
<tr>
<td>Headquarters</td>
<td>False</td>
<td>Company</td>
</tr>
<tr>
<td>ISecureInformation</td>
<td>True</td>
<td>Company</td>
</tr>
</tbody>
</table>

#### O (Operation) Table

<table>
<thead>
<tr>
<th>operation_name</th>
<th>class_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>getPhoto</td>
<td>Person</td>
</tr>
<tr>
<td>getSoundBite</td>
<td>Person</td>
</tr>
<tr>
<td>getContactInformation</td>
<td>Person</td>
</tr>
<tr>
<td>getPersonnelRecord</td>
<td>Person</td>
</tr>
</tbody>
</table>

#### A (Attribute) Table

<table>
<thead>
<tr>
<th>attribute_name</th>
<th>class_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Department</td>
</tr>
<tr>
<td>address</td>
<td>Office</td>
</tr>
<tr>
<td>voice</td>
<td>Office</td>
</tr>
<tr>
<td>name</td>
<td>Person</td>
</tr>
<tr>
<td>employeeID</td>
<td>Person</td>
</tr>
<tr>
<td>title</td>
<td>Person</td>
</tr>
<tr>
<td>address</td>
<td>ContactInformation</td>
</tr>
<tr>
<td>taxID</td>
<td>PersonnelRecord</td>
</tr>
<tr>
<td>employmentHistory</td>
<td>PersonnelRecord</td>
</tr>
<tr>
<td>salary</td>
<td>PersonnelRecord</td>
</tr>
</tbody>
</table>
### R (Relationship) Table

<table>
<thead>
<tr>
<th>super_class</th>
<th>sub_class</th>
<th>relationship</th>
<th>diagram_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company</td>
<td>Department</td>
<td>aggregation</td>
<td>Company</td>
</tr>
<tr>
<td>Company</td>
<td>Office</td>
<td>aggregation</td>
<td>Company</td>
</tr>
<tr>
<td>Department</td>
<td>Department</td>
<td>aggregation</td>
<td>Company</td>
</tr>
<tr>
<td>Department</td>
<td>Office</td>
<td>association</td>
<td>Company</td>
</tr>
<tr>
<td>Department</td>
<td>Person</td>
<td>association</td>
<td>Company</td>
</tr>
<tr>
<td>Department</td>
<td>Person</td>
<td>association</td>
<td>Company</td>
</tr>
<tr>
<td>Office</td>
<td>Headquarters</td>
<td>generalization</td>
<td>Company</td>
</tr>
<tr>
<td>Person</td>
<td>ContactInformation</td>
<td>dependency</td>
<td>Company</td>
</tr>
<tr>
<td>Person</td>
<td>PersonnelRecord</td>
<td>dependency</td>
<td>Company</td>
</tr>
<tr>
<td>PersonnelRecord</td>
<td>ISecureInformation</td>
<td>association</td>
<td>Company</td>
</tr>
</tbody>
</table>

Fig. 3.2 Converted Database Tables - continued
3.3 UML

---

```sql
// $operation_name (operation)
// $class_name $diagram_name

SELECT O.class_name INTO $class_name
FROM O
WHERE O.operation_name = $operation_name

for each $class_name
    SELECT C.diagram_name INTO $diagram_name
    FROM C
    WHERE C.class_name = $class_name
end for
```
//  get $class_name for each $class_name
    SELECT A.attribute_name
    FROM A
    WHERE A.class_name = $class_name

    SELECT O.operation_name
    FROM O
    WHERE O.class_name = $class_name
end for

//  get $relationship, $super_class, $sub_class
SELECT R.relationship, R.super_class, R.sub_class
FROM R
WHERE R.diagram_name = $diagram_name

---
4.1 UML デザイン ナンバー

UML デザイン ナンバー は、UML デザイン ナンバー の一部を記述するためのものです。UML デザイン ナンバー は、UML デザイン ナンバー の一部を記述するためのものです。UML デザイン ナンバー は、UML デザイン ナンバー の一部を記述するためのものです。

begin
{
    // UML デザイン ナンバー
    $diagram_name = input_diagram_name();
    // UML デザイン ナンバー
    if (diagram_search == TRUE)
    {
        // UML デザイン ナンバー
        $class_name[] = select_all_class($diagram_name);
        for each $class_name  // UML デザイン ナンバー

```plaintext
{
    select_all_attribute($class_name);  // バグ
    select_all_operation($class_name);   // バグ (operation)
}
    select_all_relationship($diagram_name); // バグ
}
// バグ バグ バグ バグ バグ バグ
else
{
// $class_name  NULL バグ
    while (not class_name_set_empty())
{
    insert_class_name($class_name, $diagram_name);  // バグ
// バグ バグ バグ バグ (attribute list)
    while (not attribute_list_empty())
      insert_attribute_name($attribute_name, $class_name);  // バグ
// バグ バグ バグ バグ (operation list)
    while (not operation_list_empty())
      insert_operation_name($operation_name, $class_name);
// バグ バグ バグ バグ バグ バグ
      if (relationship_seek())
        insert_relationship($ret_class_name, $class_name, $diagram_name);
    }
}
}
end
```

4.1. クラスに属する関連性を求めるために、図形を用いてモデルを表す関連性の流れを示す（図）。UML での図形は、クラスの関係性を示すための図形であり、各クラスの関係性を示す図形を用いて、関連性の流れを示す。
Fig. 4.1 Flowchart for Storing and Retrieving Class Diagram
4.2

3.1... 3.9

Table 4.1 Table Structure for Storing Diagrams

<table>
<thead>
<tr>
<th>Diagram_ID</th>
<th>int</th>
<th>4</th>
<th>×</th>
<th>ID</th>
<th>1</th>
<th>1</th>
<th>P-K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diagram_Name</td>
<td>varchar</td>
<td>50</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Diagram_Info</td>
<td>varchar</td>
<td>50</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Diagram_Owner</td>
<td>char</td>
<td>10</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td>F-K</td>
</tr>
<tr>
<td>Diagram_FDate</td>
<td>datetime</td>
<td>8</td>
<td>×</td>
<td>getdate()</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Allow_Others</td>
<td>bit</td>
<td>1</td>
<td>×</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4.1... 4.9

Table 4.1 Table Structure for Storing Diagrams
### 4.2 Table Structure for Storing Classes: Class

<table>
<thead>
<tr>
<th></th>
<th>Type</th>
<th>Length</th>
<th>Null</th>
<th>ID</th>
<th>F_K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class_ID</td>
<td>int</td>
<td>4</td>
<td>√</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Class_Name</td>
<td>varchar</td>
<td>50</td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class_Info</td>
<td>varchar</td>
<td>50</td>
<td>○</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Diagram_ID</td>
<td>int</td>
<td>4</td>
<td>×</td>
<td></td>
<td>F-K</td>
</tr>
<tr>
<td>Is_Interface</td>
<td>bit</td>
<td>1</td>
<td>×</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Stereotype</td>
<td>varchar</td>
<td>20</td>
<td>○</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Start_X</td>
<td>int</td>
<td>4</td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Start_Y</td>
<td>int</td>
<td>4</td>
<td>×</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(foreign key)
### 4.3 属性表：Attribute

**Table 4.3 Table Structure for Storing Attributes**

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Null</th>
<th>ID</th>
<th>1</th>
<th>1</th>
<th>P-K</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Attr_ID</td>
<td>int</td>
<td>4</td>
<td>×</td>
<td>ID</td>
<td>1</td>
<td>1</td>
<td>P-K</td>
</tr>
<tr>
<td>2</td>
<td>Attr_Name</td>
<td>varchar</td>
<td>50</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Attr_Visibility</td>
<td>int</td>
<td>4</td>
<td>○</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Attr_Type</td>
<td>varchar</td>
<td>20</td>
<td>○</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Attr_Stereotype</td>
<td>varchar</td>
<td>20</td>
<td>○</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Class_ID</td>
<td>int</td>
<td>4</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td>F-K</td>
</tr>
<tr>
<td>7</td>
<td>Attr_Info</td>
<td>varchar</td>
<td>50</td>
<td>○</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 4.4 操作表：Operation

**Table 4.4 Table Structure for Storing Operations**

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Null</th>
<th>ID</th>
<th>1</th>
<th>1</th>
<th>P-K</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Oper_ID</td>
<td>int</td>
<td>4</td>
<td>×</td>
<td>ID</td>
<td>1</td>
<td>1</td>
<td>P-K</td>
</tr>
<tr>
<td>2</td>
<td>Oper_Name</td>
<td>varchar</td>
<td>50</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Oper_Type</td>
<td>varchar</td>
<td>20</td>
<td>○</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Oper_Para</td>
<td>varchar</td>
<td>50</td>
<td>○</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Oper_Visibility</td>
<td>int</td>
<td>4</td>
<td>○</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Oper_stereotype</td>
<td>varchar</td>
<td>20</td>
<td>○</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Class_ID</td>
<td>int</td>
<td>4</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td>F-K</td>
</tr>
<tr>
<td>8</td>
<td>Oper_Info</td>
<td>varchar</td>
<td>50</td>
<td>○</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 4.5 Table Structure for Storing Relationships : Relationship

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Null</th>
<th>ID</th>
<th>1</th>
<th>1</th>
<th>P - K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relationship_ID</td>
<td>int</td>
<td>4</td>
<td>×</td>
<td>ID</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Super_Class</td>
<td>int</td>
<td>4</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sub_Class</td>
<td>int</td>
<td>4</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Relationship</td>
<td>int</td>
<td>4</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Relationship_Info</td>
<td>varchar</td>
<td>50</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Super_multi</td>
<td>char</td>
<td>10</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sub_multi</td>
<td>char</td>
<td>10</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Super_Rolename</td>
<td>varchar</td>
<td>50</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sub_Rolename</td>
<td>varchar</td>
<td>50</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### 4.5 Relationship (Association) : Relationship

- 29 -
### Table 4.7 Table Structure for Storing Constraints

<table>
<thead>
<tr>
<th>Constraints_ID</th>
<th>int</th>
<th>4</th>
<th>×</th>
<th>ID</th>
<th>1</th>
<th>1</th>
<th>P-K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relation_X</td>
<td>int</td>
<td>4</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td>F-K</td>
</tr>
<tr>
<td>Relation_Y</td>
<td>int</td>
<td>4</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td>F-K</td>
</tr>
<tr>
<td>Constraints_Name</td>
<td>varchar</td>
<td>20</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Constraints_Info</td>
<td>varchar</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 4.8 Table Structure for Storing Notes

<table>
<thead>
<tr>
<th>Note_ID</th>
<th>int</th>
<th>4</th>
<th>×</th>
<th>ID</th>
<th>1</th>
<th>1</th>
<th>P-K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note_Text</td>
<td>varchar</td>
<td>50</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Note_Flag</td>
<td>int</td>
<td>4</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Start_X</td>
<td>int</td>
<td>4</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td>X, Y</td>
</tr>
<tr>
<td>Start_Y</td>
<td>int</td>
<td>4</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Diagram_ID</td>
<td>int</td>
<td>4</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td>F-K</td>
</tr>
</tbody>
</table>

4.7 表示用于存储约束的表结构。3.1 表示“member”、“manager”、“(subset)”用于存储数据。

4.8 表示用于存储注释的表结构。
Table 4.9  Table Structure for Storing Links : Link

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Null</th>
<th>ID</th>
<th>F - K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link_ID</td>
<td>int</td>
<td>4</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Class_ID</td>
<td>int</td>
<td>4</td>
<td>X</td>
<td>F - K</td>
</tr>
<tr>
<td>Note_ID</td>
<td>int</td>
<td>4</td>
<td>X</td>
<td>F - K</td>
</tr>
<tr>
<td>Diagram_ID</td>
<td>int</td>
<td>4</td>
<td>X</td>
<td>F - K</td>
</tr>
</tbody>
</table>
Fig. 4.2 E-R Diagram of Relational Tables for Managing Class Diagram
4.3 企業データベース

企業データベースは、多様なデータを、関連性の高い形で管理するためのシステムです。具体的には、Microsoft SQL Server 7.0、Visual C++ 6.0、ODBC（Open Database Connectivity）を使用しています。ODBCは、データベースとアプリケーション間のコネクティビティを提供し、単一のトランザクション（transaction）内でのデータの 수정や追加が可能となります。ODBCは、制御情報の流れを提供することで、アプリケーションのインテグレーションが可能となります。
Fig. 4.3 An Example of Implemented Program
4.4 UML and its Tool Support

Rose® Plastic® UML® tools have multiple features, such as a rich graphical notation, a visual editing environment, and comprehensive documentation. However, Rose® Plastic® UML® tools have some limitations, such as a lack of support for certain modeling languages and a limited number of plug-ins. Table 4.10 shows the comparison between Rose®, Plastic®, and the proposed system.

<table>
<thead>
<tr>
<th></th>
<th>Rational Rose 2000 Enterprise</th>
<th>PLASTIC 3.0</th>
<th>Proposed System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature</td>
<td>9</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>Feature</td>
<td>○</td>
<td>○</td>
<td>×</td>
</tr>
<tr>
<td>Feature</td>
<td>○</td>
<td>○</td>
<td>×</td>
</tr>
<tr>
<td>Feature</td>
<td>×</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Feature</td>
<td>×</td>
<td>×</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.10 Comparison between Rose, Plastic and proposed system
5  36  8

UMLは、ソフトウェア・エンジニアリングにおいて、システムの導入と導出の主要なツールの一つです。具体的には、システムの設計と実装のためのメカニズムを提供します。UMLは、システム・リソースの導入と導出を可能にすることにより、進行工程（forward engineering）と逆進工程（reverse engineering）が可能になります。OODB（Object Oriented Database）は、UMLによるシステム開発において、重要な役割を果たします。


[5] http://www.omg.org - OMG home page, specification for UML and related modeling standards, such as MOF and XMI.


[16] ²►¹, ²►¹, “UML (Unified Modeling Language)” , 2000 ¾Å°ÄÁº¬, ² 79- 82, ² 2000. 10.

[17] ²►¹, ²►¹ ¹ÚÈÞÂù, ¹ÚÈÞÂù, 1998.