An Object-Oriented Design of a Multimedia Item Pool

Wang Yue Dong Tiansi Xu Liangxian Lu ruzhan
Department of Computer Science and Engineering
Shanghai Jiaotong University, Shanghai 200030, P. R. China
wy70228@mail1.sjtu.edu.cn

Abstract

An item pool with multimedia data is a fundamental part of our project, Computerized Adaptive Test System of Chinese Proficiency. This paper intends to present a design of a multimedia item pool from an object-oriented perspective. It is stressed that OO approach is especially effective in modeling and managing complex multimedia data. We first introduce our system architecture using OO-layer method. Multimedia data organization of the database is investigated and detailed class definition is given next. The object-oriented SQL-like syntax of the query language is also simply described.

1. Introduction

In our project, we need to construct an item pool to manage the items used in computerized test, in which test items are displayed on computer screen one by one for testees to answer. Unlike items in Paper & Pen test, items administered in computerized test may contain image, animation, audio and video materials, along with conventional text materials. For example, when administering listening comprehension test, the test system may first show a picture on the screen to suggest the situation where the dialogue happen, then testees can hear the dialogue and the questions through earphone, finally several choices will be displayed on the screen for testees to choose. Therefore, an item consists of media of several kinds. And the desired item pool should be able to store and manipulate all these media effectively and to integrate various types of data into a single database so that users can interact with it without knowing the difference of data in types and operations.

Due to the heterogeneous nature of multimedia data, systems intended to store, transport, display and in general manage such data must have considerably more capabilities than conventional information management system. In this paper, we shall elaborate on issues pertaining to how to design a database with multimedia data in object-oriented way.

2. Why object-oriented?

Conventional relational database system cannot manipulate complex objects effectively. It does not support user-defined data type, and its poor modeling capability makes it hard to construct complex object system. While introducing object-oriented conception, the systems are empowered with powerful data type definition facility, and object features like encapsulation and inheritance are provided. The uniform object operation greatly improves software productivity, quality and reusability. Unfortunately, object-oriented database is still immature in many ways, such as no general-purpose query language support and its incompatibility with conventional relational database. After all, OO method is comparably

0-7695-0393-4/99 $10.00 © 1999 IEEE

more suitable for our desired system.

3. System architecture

There are several methods to construct an OODB on the base of RDB, namely Gateway method, OO-layer method and Unification method. We shall concentrate on OO-layer method in this paper, because it is appropriate for our project and has more technological support for implementation. For more information about other methods, please refer to [6].

According to OO-layer method, we propose the system architecture in Figure 1.

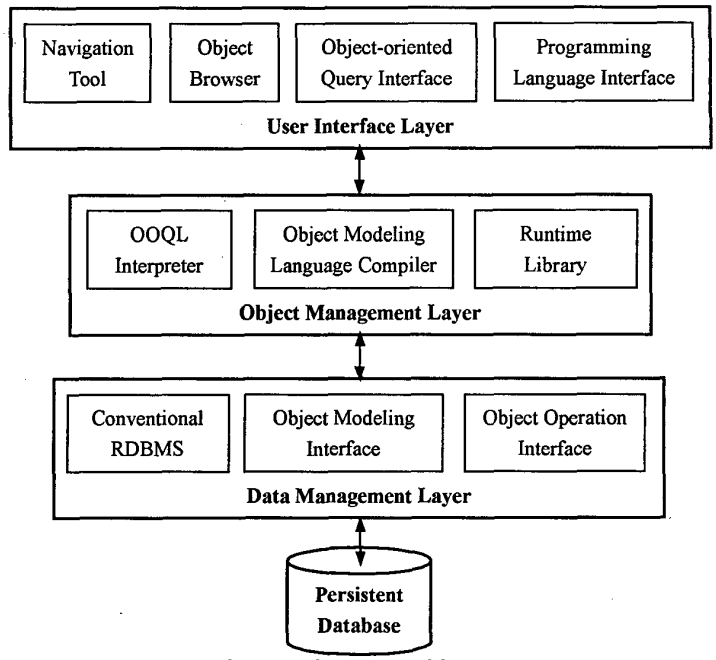

Figure 1. System architecture

As shown in Figure 1, we add an OO-layer called *Object Management Layer* outside the conventional RDB engine to make the entire system appear like an OODB to the end user. *Data Management Layer* is in essential an extended relational DBMS, along with Object Modeling Interface and Object Operation Interface, which provides operation interface for nested relations and tuples to support the complex object modeling. *Object Management Layer* just works like an object-oriented shell, having conventional RDBMS wrapped in it to make all the complex object operations transparent to end users. *User Interface Layer* handles the service requests and queries of the end user and, on the other hand, provides object-oriented programming interfaces to application programmers. In the desired system, persistent objects are efficiently stored and accessed by Data Management Layer. In general, object operations are compiled into those of the Data Management Layer by object language compiler and executed efficiently with runtime library support.

4. Organization of the multimedia data

As to the organization of the multimedia data, we introduce a method called *Referred Object Integration Method*. As shown in Figure 2, we recognize each item as an individual object of the *CItem* class. All the multimedia components in items, such as image, audio, video and formatted text, are viewed as different objects that stored respectively in several databases. The class definition will be given in the next section. In order to show that an item, say *Item1*, consists of a picture and an audio clip, a link is spanned from object *Item1* to object *Picture1* and another to object *Audio1* to denote the reference relation. The link is called *Reference Link* or *Multimedia Link*.

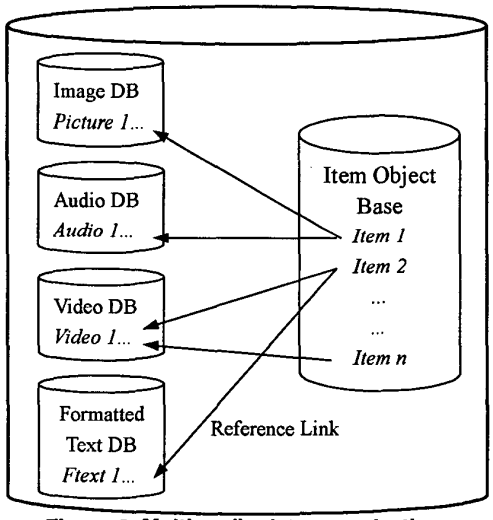

Figure 2. Multimedia data organization

By constructing link like this, the object semantic in databases can be easily understood and the database integrity can be easily maintained. Suppose we would like to delete some useless items from the item object database, it will be very convenient to find the related multimedia components by following the reference links. Moreover it is also feasible to integrate additional database organized in different media by adding new reference link to the item object database.

5. Class definition of the multimedia object

Here we denote four basic media types frequently used in our test item pool. All the class definitions are given in pseudo C++ code. Class implementation is omitted to save the length of this paper.

First we define some elementary data types to make our class definition more easily understood.

OID is a data type denoting the unique identification of a specific object.

FONT is a data type denoting character font object, including font face and size.

RECT is a data type denoting a rectangle area.

As shown in following definition, class *CImage*, *CAudio* and *CVideo* denote the class of image, audio and video respectively. General properties include object identification, storage format and size or play time. Main methods for attaching, retrieving and playing these multimedia objects are also defined. Other objects such as animation can be denoted as a series of object CImage displaying at given intervals.

```
/* Class definition of Image object */
class CImage  {
    public:
      OID    ObjectID;
      RECT   *PicSize;
      string PicFormat;     /* .bmp, .gif, .jpg, .pcx */
      ……
    /* methods */
      void   DisplayPic(OID PicOid);
      void   StretchPic (OID PicOid);
      ……
}
/* Class definition of Audio object */
class CAudio  {
    public:
      OID    ObjectID;
      long   AudTime;
      string AudFormat;     /* .mid, .wav, .mp3 */
      string AudScript;     /* Tape Script */
      ……
    /* methods */
      void   PlayAud(OID AudOid);
      void   AttachAud(OID AudOid);
      ……
}
/* Class definition of Video object */
class CVideo  {
    public:
      OID    ObjectID;
      long   VidTime;
      string VidFormat;     /* .mpeg, .dat, .avi */
      ……
    /* methods */
      void   PlayVid(OID VidOid);
      void   AttachVid(OID VidOid);
      ……
}
```

Class *CFormatText* denotes the formatted text including font face, size and other layout information, which is similar with the newly introduced data type, *Formatted Text Memo*, in Paradox for Windows.

```
/* Class definition of Formatted Text object */
class CFormatText {
    public:
      OID    ObjectID;
```

```
    FONT  Font;
    string  TxtFormat;
    string  Script;        /* plain text content */
    ......                 /* other layout information */
  /* methods */
    void    DisplayTxt(OID TxtOid);
    OID     RetrieveTxt(OID TxtOid);
    ......
}
```

Class *CItem* denotes the class of item object including general parameters describing item basic property, statistical parameters for monitoring the test process and the component link to integrate the multimedia data. Consequently, the class CItem is defined as following:

```
/* Class definition of Item object */
class CItem {
    public:
      OID       ObjectID;
    /* Basic Item Parameter */
      int       ItemID;
      string    ItemType;
      string    AbilityType;
      string    Key;
    /* Multimedia Component Link, NULL for nothing */
      CFormatText*  FText_Comp;
      CImage*       Image_Comp;
      CAudio*       Audio_Comp;
      CVideo*       Video_Comp;
    private:
    /* Inner State Property */
      int       ReferenceTimes;
      string    ContentElement;
    /* Statistic Parameter */
      real      DiffFactor;
      real      DicrFactor;
      real      GuessFactor;
    /* methods */
    public:
      OID     DefItemObj (OID image, OID audio, OID video, OID ftext);
      OID     InitItemObj (OID image, OID audio, OID video, OID ftext);
      OID     ShowItemObj (OID image, OID audio, OID video, OID ftext);
      ......      /* return the object ID of the item */
      int     ModifyReferenceTimes(OID ItemOid, int NewReferTimes);
      ......
}
```

6. User interface layer

The desired system should support two kinds of user interface, including ad hoc query for end user and application interface for programmer. What's more, graphic interface, such as *Navigator Tool* and *Object Browser* (see Figure 1), may be more beneficial to some users.

To handle the ad hoc query of end user, the system provides an SQL-like query language using object-oriented method, which we call OOQL. For example, the following query finds out the item type of the items that refer no video clips.

```
SELECT    i.ItemType
FROM      Items i
WHERE     i.Video_Comp = NULL
```

Here is another query to get the picture format and size of the item whose ID is *id*.

```
SELECT    i.Image_Comp->PicFormat, i.Image_Comp->PicSize
FROM      Items i
WHERE     i.ItemID = id
```

The OOQL statements given above will be interpreted into standard SQL statements by the *OOQL interpreter* in Object Management Layer (see Figure 1), and the Data Management Layer will finally handle the SQL query to send out the results.

For application programmers, User Interface Layer provides programming interface with some widely used object-oriented language, such as C++. Programmers can access and refer the objects in various databases in a uniform way. Because all the complex object operations are transparent to them, they can easily build applications above the User Interface Layer.

7. Conclusion

It is no doubt that multimedia database technology will be the research subject in the near future and OO method seems to be the most promising approach to manage such complex and bulky data. In this paper, we have proposed a simple design to construct an OODB for a specific multimedia item pool. The multimedia data management function of our item pool is relatively simple and its implementation is base on the conventional relational database. As in our design, multimedia data are kept in external files and the object databases only manage the file information and keep the reference link in order to integrate several kinds of multimedia data. In this sense, the system under development is not a so-called multimedia database. Now we are working on the researches of the implementation of a real multimedia database.

8. Reference

[1] Wolfgang Klas, et al., "Using an Object-oriented Approach to Model Multimedia Data", Computer Communications, Elsevier, New York, Vol.13, No.4, 1990, pp. 204-216.

[2] Yoshifumi Masunaga, "Design Issues of OMEGA: An Object Oriented Multimedia Database Management System", Journal of Information Processing, Information Processing Society of Japan, Vol.14, No.1, 1991, pp. 60-74.

[3] Francois Bancilhon, "Object Databases", ACM Computing Surveys, ACM press, New York, Vol. 28, No. 1, March 1996, pp. 137-140.

[4] Arif Ghafoor, "Multimedia Database Management Systems", ACM Computing Surveys, ACM press, New York, Vol. 27, No. 4, December 1995, pp. 593-598.

[5] Hiroshi Ishikawa, et al., "The Model, Language and Implementation of an Object-oriented Multimedia Knowledge Base Management System", ACM Transactions on Database Systems, ACM press, New York, Vol.18, No.1, March 1993, pp. 1-50.

[6] Che Dunren, and Zhou Lizhu, "Integrate Relational Database and Object-oriented Database", Journal of Software, Science Press, Beijing, Vol.7, No.11, 1996, pp. 669-675.

[7] Zheng Qinghua, et al., "Main Problems, Researching Contents and Implementation for Multimedia Database", Computer Engineering and Application, Computer Engineering and Application Press, Beijing, Vol. 34, No.1, 1998, pp. 1-4.