

**Project Title :**

**CORBA Implementation on Association Service**

**(Term Report of Course CSC 7260)**

**Student Name: Chow Kwok Chung**

**Student ID: 96193960**

**PTMSC**

---



---

# Table of Content

<b>ABSTRACT .....</b>	<b>5</b>
<b>1. INTRODUCTION .....</b>	<b>6</b>
1.1 ASSOCIATION SERVICE.....	6
1.2 ASSOCIATION LIST .....	7
1.3 HASH TABLE.....	7
1.4 PLATFORM RUNNING ASSOCIATION SERVICE .....	8
1.5 COMPARISON BETWEEN CORBA AND DCOM .....	9
<b>2. DISTRIBUTED COMPUTER SYSTEM AND CENTRALIZED COMPUTER SYSTEM. 11</b>	
2.1 CENTRALIZE COMPUTER SYSTEM .....	11
2.2 DISTRIBUTED COMPUTER SYSTEM .....	12
2.3 DIFFERENCE BETWEEN THE CENTRALIZED COMPUTER SYSTEM AND DISTRIBUTED COMPUTER SYSTEM: .....	13
<b>3. OVERVIEW OF CORBA .....</b>	<b>18</b>
3.1 CORBA STANDARD .....	18
3.2 CORBA IMPLEMENTATION .....	21
3.3 OVERVIEW OF JAVA ORB.....	22
3.4 PROGRAMMING STEP.....	23
<b>4. DESIGN OF SYSTEM .....</b>	<b>24</b>
4.1 STRUCTURE OF ASSOCIATIONS .....	24
4.1.1 Access Control.....	25
4.1.1.1 Capability.....	27
4.1.2 tuple.....	29
4.1.3 List structure.....	30
4.1.4 Hash Table.....	31
4.1.5 Storage of associations.....	32
4.1.5.1 Approach 1 : storing in a single file .....	34
4.1.5.1.2 Recoverable Associations .....	35
4.1.5.2 Approach 2 : storing in database .....	36
4.2 PROGRAM'S MODULES .....	40
4.2.1 tuple module:.....	40
4.2.2 List module:.....	40
4.2.3 ListHandler module: .....	41
4.2.4 HashTable Module .....	43

---



---

4.2.5	<i>tableHandler Module</i> .....	43
4.2.6	<i>Capability Module</i> .....	45
4.2.7	<i>Association Module</i> .....	47
4.2.8	<i>Client Module</i> .....	51
4.2.8.1	<i>New Capacity (Get New ID)</i> .....	52
4.2.8.2	<i>Specify "user name" and "password"</i> .....	52
4.2.8.3	<i>Put Association</i> .....	53
4.2.8.4	<i>Get Association</i> .....	54
4.2.8.5	<i>Delete Association</i> .....	55
4.2.8.6	<i>Restricted Capacity (Get Restricted ID)</i> .....	56
4.2.8.7	<i>Save the operations on associations</i> .....	57
4.3	<b>SYSTEM DEVELOPMENT</b> .....	57
4.3.1	<i>Phase I: a single program</i> .....	58
4.3.2	<i>Phase II : Construct a program in Client-Server based approach</i> .....	59
4.3.2.1	<i>Step 1) Define IDL</i> .....	60
4.3.2.2	<i>Step 2) Implement these interfaces with Java class</i> .....	63
4.3.2.3	<i>Step 3) Object Implementations</i> .....	66
4.3.2.4	<i>Step 4) Creation of Client program</i> .....	66
4.3.2.5	<i>Step 5) Run the Client/Server Program</i> .....	67
4.3.3	<i>Phase III: Permanent and Recoverable Attributes of the Associations</i> .....	68
4.3.3.1	<i>Steps to involve the database via JDBC program:</i> .....	71
<b>5.</b>	<b>SYSTEM REQUIREMENT</b> .....	<b>73</b>
<b>6.</b>	<b>EXPERIENCE FOR THE DEVELOPMENT OF THE PROJECT</b> .....	<b>74</b>
6.1	USAGE OF PROGRAMMING LANGUAGE.....	74
6.2	APPLICATION AND APPLET .....	74
6.3	DATA RETRIEVAL FROM DATABASE.....	75
6.4	INDEX ON DATABASE .....	77
6.5	LIMITATION OF THE SYSTEM .....	78
<b>7.</b>	<b>FURTHER DEVELOPMENT</b> .....	<b>80</b>
7.1	DEVELOP CLIENT SIDE OF THE SYSTEM AS AN APPLET .....	80
7.2	ADDING MORE INTERFACE FOR VARIOUS FORMAT OF "VALUE" .....	81
7.3	MULTI-USERS SUPPORTING .....	81
7.4	DIRECTORY SERVICE AS A YELLOW PAGE.....	81
<b>8.</b>	<b>CONCLUSION</b> .....	<b>83</b>
	<b>REFERENCE</b> .....	<b>84</b>

---



---

**APPENDIX I - SYSTEM INSTALLATION GUIDE .....87**

**APPENDIX II - USER GUIDE .....89**

## **Abstract**

Association service is a software entity running in a distributed computer system to provide a service on a mapping between a key and a value. The "key" is in form of a string and the "value" is a sequence of bytes. In other words, the "value" may be a simple text or a binary data such as image or audio.

It applies CORBA standard and the Visigenic's VisiBroker for CORBA Implementation in client-server based approach. The design of the system is broken into several phases. During the development of the system, the first step is to create a single program, which consisted of a number of modules, running on a single machine. Afterwards, the single program is broken down to be client side program and sever side program, both of them are ported to the distributed computer system and linked by Object Request Broker (ORB). More and more features have been added step by step. The final version of association service has included capability for access control and the associations with permanent and recoverable attributes.

# 1. Introduction

## 1.1 Association Service

Association is simply described as just a mapping from a key to a value. A service is a software entity running on one or more machines. In other words, an association service is a service provided in a client-server based system that a server should return a requested value to its caller whenever a key is submitted from a client side if the mapping is successful. Otherwise an error message will be returned.

Because of the simple definition of an association, there are many applications may be regarded as a kind of association service. For example, the design on telecommunication interfaces for deaf people [1], an on-line grammar learning service or on-line dictionary look up service could be considered an association service.

There are different mechanisms to create a link between a key and a value (it is also called datum) [2], such as association lists, "one-dimensional" tables, the association table, Hash table, Balanced binary trees, Red-Black trees and Weighted Balanced tree. Besides, it may apply the database to hold "keys" and their corresponding "values" together for their permanent storage.

In this project, it has applied the association lists and hash table for searching a tuple and storing a tuple in volatile memory. It has also used the database for permanent storage of the associations.

## 1.2 Association List

An association list is a data structure used very frequently for Scheming. It is a list of pairs of values, each of which is called an association or so called a node of the list.

One of the advantages of an association list [2] representation is that the length of the list is flexible and could be incrementally augmented simply by adding new entries to the front of the list.

In case of the sequential searching of the list, new entries can "shadow" the old one. If a list is viewed as a mapping from keys to data, then the mapping can be not only augmented but also altered in a non-destructive manner by adding new entries to the front of the list.

However, the average lookup time for the searching on an association list has linear relationship with the number of associations. The speed of seeking will be slower down when there is an increase in the number of associations existing in the system.

In order to speed up the searching, the association lists will combine with the mechanism of hash table in this project.

## 1.3 Hash Table

Hash table, which applies an algorithm of hashing function for indexing, is a powerful mechanism with constant-time accession to large amount of data. Hash table defined in this project is not as flexible as an association list, but because its access time is independent of the number of associations in the table. The average time spent on the

---

---

insertion, deletion, and lookup operations is bounded by a constant. In case of that, The use of hash table is popular for most applications. To build up a hash table, it needs a hash function.

A perfect hashing function [14] is minimal if the integers onto which a particular list of words is mapped to form a contiguous set that is a set without holes.

Minimal perfect hashing function are useful in applications where a predetermined set of high frequency words is expected and the hash value is to be used to index an array related to those words. If the hashing function is minimal, no elements in the array are wasted.

#### **1.4 Platform running association service**

The association service may be running on a single machine or it may also be running in a large scale of computer system that consists of tens of different machines. However, distributed computer system is much complicated compared with the architecture of a single machine running with a single operating system.

In order to simplify programmers' jobs, some frameworks are created as the role of middleware for the design and implementation of new applications in the distributed computer system. The Remote Procedure Call (RPC), which was introduced by Sun Microsystems at the beginning of the 1980, is one of ways to develop the association service [11]. The RPC was a widely used method [7] at that period. However, it is a standard of procedural approach.

Over the last decade, object-oriented development methods and techniques have been

---

---



much popular. In response to the market, two standardized commercial architectures, CORBA and Microsoft's DCOM (Distributed Component Object Model), both of which support the object-oriented programming, are developed for helping the construction of application in a distributed system. And therefore, the position of RPC has been replaced by them.

## **1.5 Comparison between CORBA and DCOM**

- **Similarity**

There are more similarities between CORBA and DCOM than their differences [20].

One common characteristic between both systems is the use of Interface Definition Language (IDL).

Both CORBA and DCOM use IDL to define only the interface of an object, which may be implemented in any language such as C++ or Java. The separation of interface from implementation makes IDL be the significant aids to software construction.

Both architectures address the problem of distributed computing through the use of a client/server base. Their IDL compiler automatically provides all the middleware needed to locate and connect the client and server objects. What the developer needs to do is only the generation of the client and server code.

Another common feature of CORBA and DCOM is that the distributed servers is location transparency for the accession from client program.

- **Differences**

Both architectures provide a means that allows server and interface information to be

---

---

stored on disk, but their storing methods are not same. While DCOM makes use of the operating system registry for the server information and Type Libraries for interface information, CORBA implementations maintain files under the ORB's implementation directory.

Another major difference between CORBA and DCOM is the method for error handling. DCOM makes use of returned values in a structure called an HRESULT whereas the exception for error handling is used by CORBA.

Most of DCOM's infrastructure is defined and implemented. On the contrary, a large portion of CORBA's has not been implemented yet. It is much easy for version control on DCOM because its interfaces are immutable and so the interface to the server will be constant when the client is compiled.

This project has selected the CORBA standard and Visigenic's VisiBroker for its Implementation based on the consideration of the availability of the system and updated technology.

---

---

## 2. Distributed Computer System and Centralized Computer System

### 2.1 Centralize Computer System

Centralized Computer System [3] is to place the main computer at the focal point of all data process services (ref. to figure 2.1). In a centralized approach, online communications pass through a central computer, which also controls access to the system's files. The advantages of a centralized approach are simplicity, low cost, elimination of duplicate computer hardware, and efficient use of processing resources. And therefore the mainframe computer connected with several terminals was basic structure of computer system decades of years ago. But in recent years, the Distributed Computer System has already replaced the leading position of Centralized Computer System.

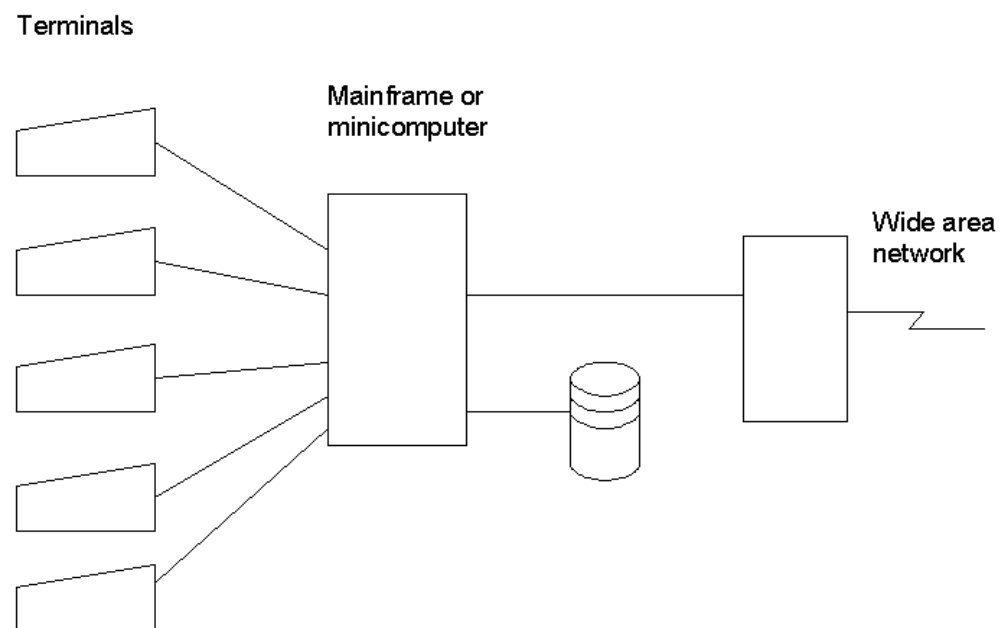


Figure 2.1

## 2.2 Distributed Computer System

A Distributed Computer System (ref. to figure 2.2) is a collection of processor-memory pairs connected by communications sub-networks and logically integrated in varying degrees by a distributed operating system and/or distributed database system [4]. A distributed computer system is a universal set of the physical network and all the controlling software. Local area networks, wide area networks, network operating systems, distributed real-time systems, and distributed databases are just particular instance of a Distributed Computer System but not all.

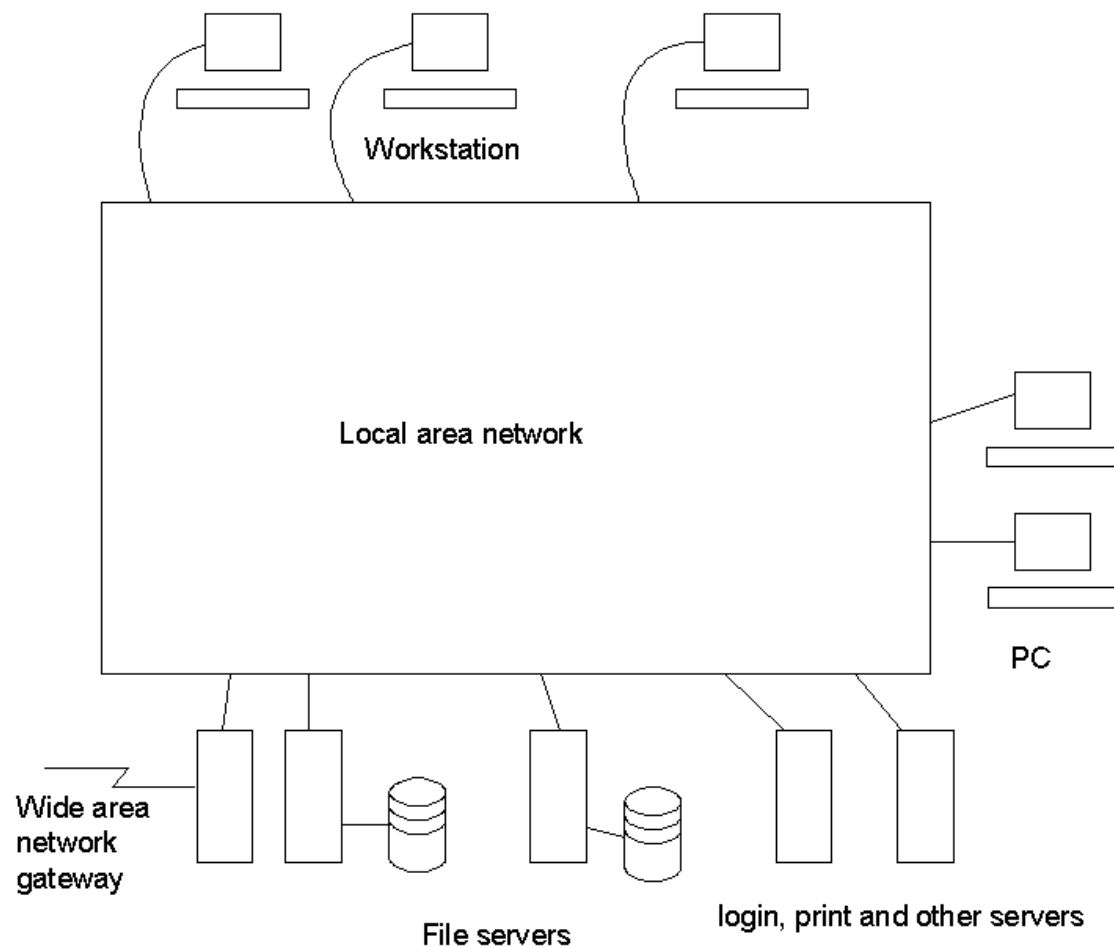


Figure 2.2

There are a number of reasons for the popularity of Distributed Computer Network [5]:

- It needs to integrate the existing computer systems. The world phone system is one of examples.
- It needs to fulfil the requirement of some applications, which will spend a large volume of computer capacity.
- It may reduce the communications' costs.
- It is more availability because of a failure limited in a single site.

### **2.3 Difference between the centralized computer system and**

#### **distributed computer system:**

The set of abstractions that the centralized operating system provides for use in application programs is determined by a single immutable interface. On the contrast, the application programs running in distributed computer system can access many different services, each of them providing its own procedural interface for accessing resources. The resources for application programs can be extended by adding new services.

In centralized computer system, the operating system is the main system software layer. It manages all of the basics in the system and provides essential services to

---

---

application programmers and users. Most of services are performed primarily by the operation system kernel in centralized computer system. The layers in centralized computer system are connected one by one in serial order. Applications <-> Programming Language Support <-> Operating System<-> Hardware. All of the processor and memory resources are available for allocation by the operating system in a centralized computer system.

On the contrary, The structure of the distributed computer system is in different features. The role of the kernel is restricted to only the basic resource management including memory allocation and protection, process creation and processor scheduling, inter-process communication and peripheral device handling. All other shared resources and services are handled by "open services", which is a new class of software components in a distributed computer system. Any services that do not require privileged access to the kernel's code and data or the hardware of the computer need not be included in the kernel.

There are six key characteristics [11] are primarily responsible for the usefulness of distributed systems:

### **1) Resource sharing:**

The resource may be hardware components such as disks and printers or software components such as files or databases. It is useful for the software developer because they always work as a team and need to share the same development tools or exchange their developing modules.

### **2) Openness**

---

---

The openness of a computer system is the characteristic that determines whether the system can be extended in various ways. The open computer system may be extended at the hardware level by the addition of computers to the network and at the software level by the introduction of new services. This characteristic enables application programs to share resources of the system.

### **3) Concurrency**

The executions are in concurrency if there are several processes existing in a single computer. In distributed systems there are many computers connected together and so it can run the processes in parallel. Many users may invoke commands simultaneously and many servers can respond to different requests sent by client processes concurrently.

### **4) Transparency**

Transparency is that there is the concealment for the separation of components from the user and the application programmer. And therefore the users may work in the distributed environment as if running on the same machine. The skill to access remote resource is indifferent from those for local. There are different kinds of transparency:

- access transparency,
  - location transparency,
  - concurrency transparency,
  - replication transparency,
  - failure transparency,
  - migration transparency,
- 
-

- performance transparency,
- scaling transparency.

### **5) Scalability**

Distributed systems can be operated at many different scales. It may be just a small network or interconnect with many small networks to be a big network. For example, internet is a very big network consisting of many smaller networks.

### **6) Fault Tolerance**

Fault Tolerance is a broad sense. Machine failure, communication faults, storage device crashes are all considered faults and should be tolerated.

The appearance of system crash may lead a great lost of business. To avoid the disaster appeared, in distributed systems, it always applies the redundancy technique for the fault tolerance. It means that the system may run normally even if some fault appears. For example, the whole banking system will not be down even if the computer system of its branch out of service.

An advantage of distributed systems over centralized systems is the potential for fault tolerance and the ability on scalability because of the multiplicity of resources. However, inappropriate design can obscure this potential and worse, hinder the system's scalability and make it failure prone. The consideration of fault tolerance and scalability calls for a design demonstrating distribution of control and data. Any centralized entity - a central controller or a central data repository, introduces both a sever point of failure and a performance bottleneck [19].

It is comparative easy for a programmer to develop an application in centralized

---

---



computer system compared with that in distributed computer system. It is because there are a number of additional features need to be handle in the distributed system.

In comparison with those in a single stable platform using a single operating system and a single programming language, the design and implementation of a software is much difficult and complex in distributed system.

The additional complexities [6] that should be faced are listed as the following:

- It must run on a network of machines. The distributed components of the system should satisfy the overall target of the software.
- The machines on the system may run different operating systems, such as Unix, MS Windows and many others.
- The components of the system should be capable to integrate into new system even it has no plan before.
- It should be permitted to use different programming language. For example, Java may be used at the front end whereas C++ is applied for core functionality.

### 3. Overview of CORBA

#### 3.1 CORBA Standard

CORBA is an abbreviation of "The Common Object Request Broker Architecture". It is an important standard for distributed object oriented systems [6,8]. It is defined by the Object Management Group (OMG) - an open consortium of 300 companies and institutions ranging from system providers to users. Its aim is to provide support for the use and management of objects in a distributed system of heterogeneous machines and operating systems.

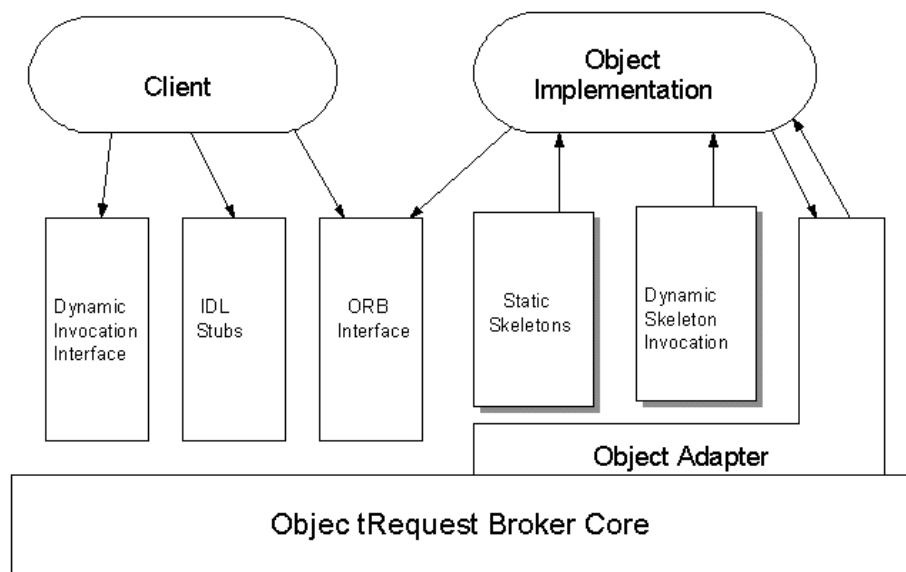


figure 3.1

#### **ORB**

The central component of the CORBA system is the Object Request Broker (ORB), which supports remote invocations. When a client invokes an operation, the ORB is responsible for finding out a related object, delivering the request to that object and

returning the response if any to the caller (ref. to figure 3.1). In other words, a client can invoke a function of a remote object on another machine as if the invocation of a local function.

Besides an interface for communicating with the ORB from either client or server, it also needs to have a component, which is called Object Adapter, to be responsible for interaction between object implementations and the ORB.

### **Object Adapters**

An object adapter is a component that an object implementation uses to report its availability for request to an ORB while ORB uses this adapter to manage the running environment of object implementations [21] .

There are different object adapters defined against different object implementations. Currently, CORBA has defined two interfaces, the Basic Object Adapter (BOA) and Portable Object Adapter (POA). In this project, the object implementation has utilized a BOA.

### **BOA**

The BOA is a logical component of the ORB. For the object implementations, it is an interface used to inform the ORB the existence of them and the availability of them for request by client. On the other hand, the client considers the BOA to be the component of the ORB, by which the client may invoke an object reference and interact with the object implementations.

Overall speaking, the BOA can launch processes and dispatch request to them when they have been initialized.

## **POA**

There are some vagueness on the semantics of the BOA specification such as the specific features for various platforms and the way to achieve the implementations. As a result, The implementations by different vendors are inconsistent each other. To eliminate this situation, the POA is developed to standardize some of the proprietary features.

The POA is used to provide a large set of interfaces for managing object references and their implementations. Consequently, The code written using the POA interfaces may have the same semantics in every ORB.

## **IDL**

All CORBA objects must have an interface to be defined via the Interface Definition Language (IDL). IDL is not considered a programming language because it cannot be used to implement the interfaces that are defined it. It can not replace the roles of the traditional programming language such as C++, JAVA or Ada.

Although IDL is not a complete programming language, it has the similar features - inheritance, higher reusability and less modification efforts, as the traditional object-oriented programming language. Accordingly, it is easy to port to such proper object-oriented languages such as C++ or JAVA for further design.

Programs built with CORBA use mainly the native programming language syntax such as C++ or JAVA, and should be portable between different platform [9].

Associations between data objects are also easier to manage through a database than between CORBA object.

Although the OMG has defined CORBAservices and the CORBAfacilities to extend

---

---

the built-in support for applications, some issues are worth careful consideration when the implementation of application objects [9]: CORBA objects tend to use more memory, and their creation and deletion may be slower than for ordinary object. Large data transfers may also be more efficient and simpler to perform via databases or shared file systems.

Accordingly, This project has utilized a database for handling the storage of associations.

### **3.2 CORBA Implementation**

CORBA is just an infrastructure for constructing distributed program in distributed computer system. It needs to have an ORB to implement the CORBA.

The IDL compiler will compile the IDL file and generate a client stub and server skeleton, which is a kind of the traditional programming language code, for further coding. The code may be the code of C++ or Java based on the kind of ORB applied.

Three popular CORBA ORBs [10] are described as the following:

- **Iona's Orbix**

Iona is the leading provider of CORBA technology. Its C++ Orbix ORB now runs on over 20 operating systems including Unix, NT, Windows 95 and etc. The latest version of its OrbixWeb can fully implement the CORBA IDL and Java mapping.

- **JavaSoft's Java IDL**

It is pure CORBA/Java ORB including an IDL-to-Java compiler to generate the

---

---

new CORBA/Java IDL mapping. It is free and also a minimalist CORBA ORB.

- **Visigenic's VisiBroker for Java**

It was the first Java IIOP ORB to support both client and server functions. The 3.0 version has enhanced the server by providing thread-pool and connection managers. The updated version is 3.4.

In this project, it has utilized the Java for coding and so Visigenic's VisiBroker for Java 3.4 is the selected ORB to implement CORBA.

### **3.3 Overview of Java ORB**

A Java ORB is an ORB that supports a Java language mapping for OMG IDL. It allows clients and object to be implemented in Java.

The simplest scenario involving Java ORB should have five components. They are two Java virtual machines (JVMs), the client program, the server program and the ORB. The client communicates with the ORB in order to transfer a request for service to the server and server will sends the reply via the ORB back to the client. Both Client and Server are running on their JVMs.

The IDL compiler generates a number of Java classes, which are grouped to be stub classes and skeleton classes just as what have been described in former section.

### **3.4 Programming Step**

To create an application with VisiBroker, it should implement as the following steps:

- 1) Specify all of the objects and their interfaces using the OMG's IDL
- 2) Implement these interfaces with Java class (using the VisiBroker idl2java Compiler to generate stub routines for the client program and skeleton code for the object implementation- server program)
- 3) Refer to the skeleton code, writing codes to create the server that implements the object.
- 4) Refer to the stub code, writing codes for the client.
- 5) The code for the client and object, once completed, is used as input to a Java compiler to produce a Java applet or application and an object server.

## 4. Design of System

### 4.1 Structure of Associations

Just as the above descriptions, the concept of an association service is a service provided for a mapping from a key to a value. The mapping may be a match between a string and another string or submission of a string with an image as a return.

In order to have flexible format of data, the "value" returned from server side is designed to be in form of a sequence of bytes while a string type the "key" is assigned to, and therefore the "value" may be used to represent a simple text or an image.

The following associations can be handled in this developed system:

- Submit a string from client side and return another string, which may be number or characters from server side.
- Submit a string from client side and return a Unicode Character, for example, a sequence of Chinese character. The system itself, however, does not have any specific function to handle the Unicode set. The ability to display Chinese Character depends on whether the operation system in client side supports this feature or not.
- Submit a string, which is the name of an image, from client side and return the bytes stream of that image, which is in jpeg format or gif format, and then invoke an interface to display the image to user.

The above mapping between a key and a value is just some selected samples for association service developed by this system. The fact is that other kinds of mapping such as audio clips may be handled if there is further design on decoder. The decoder

---

---



may be a module in Client program or another server designed for this special service. In order to simplify the design of this project, the image handler is only a module in the client program.

This system is designed to be a multi-users system. It means that various users may operate their own associations in the same server. In case of that, it should have an identifier (ID) which is unique and used to distinguish the owners of associations assigned to individual user. Based on this unique ID, the server may recognize the corresponding value as a reply to the submission of a special key. Because of the existence of unique ID, the submission of a same key but with different ID may get different value as a return from server side.

There will be a number of different users to access this system to get their own associations. Besides the consideration of the accuracy of a mapping between key and value, it should also consider the security of data in the server. In order to protect the data from unauthorized access, it needs to have some mechanism to limit the accession. In fact, access control is an important feature in the design of application in the distributed computer system.

#### **4.1.1 Access Control**

Access control is a fundamental mechanism to maintain security in computer systems. It is a process that decides who is authorized to access a specific object and who isn't. Either an access control list (ACL) approach or a capability approach is usually applied for the access control mechanism.

This project has adopted capability approach for the access control in the security issue of the association service.

Kao and Chow [12] in their paper have suggested three factors - performance, scalability, and ease of separating policies and mechanisms, to be the main reasons that capability is actually more suitable for object protection than ACL upon which authorization mechanisms of many conventional operation base in distributed system.

### **1) Performance**

In a capability base system, an object manager only needs to validate a capability for each object access but no need to have extra information of individual capability.

Therefore, the system may spend less resource to handle this issue.

On the other hand, an Access Control List system requires higher overhead for object accession. It ought to spend more resources to handle the searching and checking of an entire access control list which size could be very large, especially in an enterprise-wide distributed environment. The exhaustion of resources by the access control list will be larger and larger when there are more and more users.

### **2) Scalability**

A capability base authorization system is obviously more scalable than an ACL based authorization system in a distributed system environment, since each access authorization is completely independent of the size of a distributed system. The operating cost of each authorization based on capability is always  $O(1)$ , while the authorization based on ACL is  $O(n)$ , where  $n$  is the size of the access control list. The loading on the system will be heavier and heavier when the number of users is

---

---

increasing.

### **3) Ease of separating policies and mechanisms**

Modern operating systems usually centralize all access control policies in an authorization server. It requires all object managers should be restricted to contain access control rules and mechanisms to enforce these policies.

Distributed and local checking of capabilities by object managers is more adaptive to such an architecture and has no need to maintain a replicated copy of the whole authorization information database. The object manager is responsible for all the activities regarding capability, including generation, distribution, verification and revocation of capabilities.

#### **4.1.1.1 Capability**

A capability may be regarded as a 'digital key', which is a very large integer selected in a manner that makes it difficult to counterfeit. There are several ways to ensure that the constructed key is unique and difficult to forge.

In the ICAP architecture proposed by Li Gong[13], a pair of capabilities: internal and external capabilities are applied. A capability is a bit string and only the server can propagate a valid capability. When the server creates a new object on behalf of client C1, an internal capability is created as ( Object, Random0 ) and stored in the server's internal table. This table is protected against tampering and leakage. C1 is sent an external capability (Object, Rights, Random1) which looks exactly the same as a classic capability but  $\text{Random1} = f(\text{C1}, \text{Object}, \text{Rights}, \text{Random0})$ .

When C1 presents the capability later, the server runs the one-way function  $f$  to check

---

---

its validity. C1 should explicitly present the request to the server if it wants to pass the partial access authority to other clients. The capability of partial access authority will be in the similar format as that for C1 but replace C1 and Random1 by C2 and Random2 where  $Random2=f(C2, Object, Rights, Random0)$ .

The capability may be regarded as the password for a user to login the system to access his/her own set of associations. It is impossible for users themselves to generate unique capabilities as a password. There should be a capability server responsible for the creation of password to an individual user. Accordingly, two interfaces are provided at client side for a user to request a capability as his/her password before he/she can manipulate his/her own set of associations in the association service.

These two interfaces are:

### **1. NewCapa**

This interface lets a user generate his/her unique capability in the association service and so that he/she can manipulate the operations on associations under that capability.

### **2. RestrictCapa**

It is common that the owner of a capability does not want to disclose his/her own capability to other one or he/she only want to share only partial attributes but not all of the access authority to someone. In case of that, the RestrictCapa is a channel for the creator of a set of specific associations to propagate his/her access authority to another one.

The users may manipulate the operations on the set of authorized associations when

---

---

they have got the legal capabilities. The grouping of associations under the same ID is a concept of set. Accordingly, they are regarded as the elements of the same set.

#### **4.1.2 tuple**

Each association is defined by a tuple of values (id, key, value, size).

The main components in the tuple are "key" and "value". The roles of "key" and "value" have already been defined in previous section. The use of "id" is to identify the specific set that a tuple belongs to and the amount of "size" is used to indicate the number of bytes of the "value". The operations on tuples are the main missions of this system. Each operation may have corresponding reaction to express the status whether the request from client side may have successful reply from server side or not.

Above elements in a tuple have already provided enough information to distinguish unique set of associations. However, in case of the manipulation of different formats of values. It needs to know the format of "value" when it is returned to client side from server side. And therefore, the tuple has been added with one more variable, "valueType", which is used to indicate the format of the "value". Based on "valueType", the client side program may wake up corresponding user interface to display the specific format of "value".

Therefore, the content of tuple has included five elements - "id", "key", "value", "size" and "valueType". Their corresponding data types are listed in table 4.1.

---

---

<b>Variable Name</b>	<b>Data Type</b>
id	long integer
key	string
value	array of bytes
size	integer
valueType	string

table 4.1

### **4.1.3 List structure**

The grouping of associations is a concept of "set". In order to link up the tuples in a same set, a list structure is used to achieve this objective. The list is used to link up the tuples one by one.

Each new tuple will be added in the front of the specific list. It means that the last added tuple will be the most updated one. Based on this attribute, the update function can be achieved by the submission of another tuple with a same value of "key" but new value of "value".

It is handy to link up all tuples in a single list. However, the searching time for getting a specific tuple will be proportional to the length of the list. It will be the worst case if the requested tuple is at the end of the list.

#### 4.1.4 Hash Table

To speed up the searching operation, The tuples are stored in 256 lists, which are indexed by a hash table. The structure of hash table is so simple that it is just an array of list but indexing by a hash function, which has referred to the algorithm proposed by Peter Pearson. This algorithm may handle variable-length text strings,. If the input string consists of n characters and  $C_1, C_2, \dots, C_n$  represent each character or space of the string, the proposed algorithm will be like the following:

```
h[0] =0;
for i = 1 to n
    h[i] = T [ h[i-1] XOR C[i] ] ;
next
return h[n];
```

where XOR is an bit-wise exclusive OR operation and T is an array of n elements.

In order to assign the value of the array of T randomly, the following algorithm is also applied to generate the table T:

```
unsigned char T[N]
makeT(nPerms, seed)
    int nPerms, seed;
    {   int i, a, b, tmp;
        for(i = 0; i < N; i++) T[i]=;
            srand((unsigned) seed);    // seed for random numbers
```

```
for(i = 0; i < nPerms; i++) {  
    a = rand() % N;           // % symbol is the mod operation  
  
    b = rand() % N;  
  
    tmp = T[a];  
  
    T[a] = T[b];  
  
    T[b] = tmp;  
  
    }  
  
}
```

#### **4.1.5 Storage of associations**

The associations will be abolished and unrecoverable when the service is terminated if it is held only in the volatile memory. But it is most likely that the associations are expected to be for permanent use.

To make the persistent associations, they must be saved in disk storage to maintain the elements of tuples and their relationship even if the server has been down, and then they may be recalled for use when both the server side program and client side program are woken up again.

It is not strange that the server will be suddenly down for accident and all of manipulations on data will be damaged. To avoid this disaster, there should be a mechanism to protect the operated data and guarantee the recovery of data when the server is woken up again. In other words, associations should have permanent and recoverable attribute. All the information including the tuples, the lists and the hash

---

---



indexes should have copies in a secondary storage for recoverable besides the copies in volatile memory.

The structure of permanent data in a file system, however, is fundamentally different from those residing in the execution environment - virtual memory [15]. The additional requirement on storing permanent data in a file system of Unix environment raises up another problem. In order to output the pointer, tree, graph, and other complex data type from the volatile memory to the permanent storage, the programmer ought to construct a module to flatten them. This additional requirement will make the design of application become more complicated.

The work on "flattening" and "unflattening" the data structure will be comparative simple if the data need to be saved is a simple text document which consists of serialized strings of text. On the contrary, if the data are in complex data structure, for example pointer-based structure, it will be much difficult for the programmer to handle them.

Atkinson [16, 17] proposed that all those in a system should be able to survive for as long as the data is required; he called this attribute longevity persistence. He also proposed that all data should be treated in a uniform manner regardless of the length of time for which it persists. The persistence of data is orthogonal to its other attributes, such as size, type, and ownership. Systems providing this abstraction are said to support orthogonal persistence.

Unfortunately, Unix operation system, which is always a popular platform for the client-server based application running on, is clearly not appropriate in a persistent environment. The file system of Unix machine does not have this manner and much

---

---

of its I/O is eliminated by a single-store abstraction.

#### **4.1.5.1 Approach 1 : storing in a single file**

One of approaches to make the associations to be permanent is to create a simple file to hold all the information about the manipulation on tuples, list and hashtable. Each time when the system is restarted, all the information about associations saved in that file is read and stored in the volatile memory again.

The saving and the restoration of associations are done explicitly. A user himself/herself should determine whether the modification done on the associations is saved or not. Therefore, two functions need to be designed:

- **RetrieveAssociation:**

Read a flattened association from the file referenced by FileID, and then unflatten it to be a structure of tuple (that is id, key, value, size) and add it into the appropriate list of associations. A message will appear to indicate whether the retrieval of association is success or failure.

- **StoreAssociation:**

A FileID and a tuple in volatile memory are taken as arguments and appended to an external file. The added information in the file is for future retrieval. If there has already been a tuple with the same id and key as that of the new added tuple, the new one will override the old one in the file.

##### **4.1.5.1.1 Flattening and Unflattening Tuples**

---

---

As the previous description that the content of Unix file [18] must be a sequence of bytes, and it must have a mechanism responsible for the conversion of data structure.

Accordingly, a pair of functions ought to be defined to implement the flattening and unflattening of data:

- **StatusFlatten:**

This function is used to flatten a tuple to be an array of bytes, which will be exported to a Unix file. The number of bytes should also be included for the future unflattening.

- **StatusunFlatten:**

This function is used to re-construct the tuple when given an array of bytes, which is actually a flattened tuple.

#### **4.1.5.1.2 Recoverable Associations**

Above file operations are done explicitly. That is, the user at client side determines whether the saving or restoration of the associations is done or not. Those functions still have not handled the recovery of associations.

A recoverable association service should be transparency to the users at client side. It means that the server should make a backup for the associations for the recovery automatically.

To make all the manipulation on the associations to be recoverable from server crash, each operation done on associations must be saved once it is invoked. And so the recovery can be done immediately when it is restarted from a sudden crash.

---

---

Obviously, the approach depicted at above paragraph requires the programmer to spend much of time to handle the conversion on data structure. It is not a good method for the construction of persistent and recoverable association service. The fact is that there is another simple way to handle the storage of data.

#### **4.1.5.2 Approach 2 : storing in database**

Although the Java environment has the built-in object serialization facility [22], which converts all the objects in the network to a byte stream for the storage in a flat file, the serialization stem is limited by the fact that the entire network of objects must be access as a whole. It is not possible to manipulate an individual object. Each time a member of the object network is stored, the entire byte stream containing all objects in the network must be serialized and stored. Obviously, the approach on serialization is insufficient for sound application development. It needs a programmer to spend much time to handle such file operations described at section 4.1.5.1. Therefore, there is another approach that storing the associations in a database.

In this approach, a programmer need not spend time to handle the flattening and unflattening of data structure. All of them are handled by the database server. However, It needs to have a middleware for the connection between a Java program and a database server (ref to figure 4.1). A most popular middleware is the Java Database Connectivity (JDBC) which becomes a part of the core Java API with the release of version 1.1 of JDK.

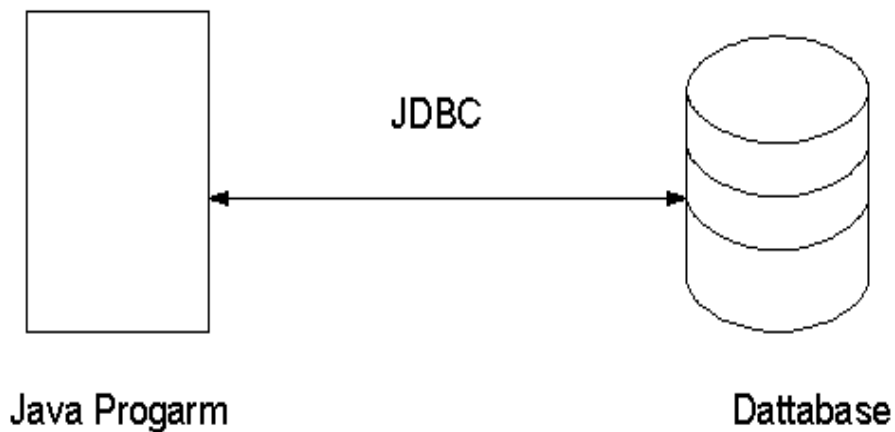


figure 4.1

The Java Database Connectivity (JDBC) is a good alternative for persistent object development with Java. It may let the access from the application program to the database server be available.

- **JDBC**

The JDBC standard adds a whole new dimension to this capable and multi-faceted language [23]. Using this API, the developer can use Java applications for standard database programming tasks such as reports or updates. It may play a role of filter program built in the application to read data from data stream, and impose the conversion before updating a database.

Java applications written using JDBC are portable both for the hardware platform and the database. The JDBC is based on ODBC, through which most of major databases can be accessible.

JDBC uses the industry standard Structured Query Language (SQL) to communicate with the database server.

- **Database Server**

A database server is the key for solving the problems of information management.

In general, a server must reliably manage a large amount of data in a multi-user environment where many users can concurrently access the same data. This requirement must be accomplished while delivering high performance. A database server must also prevent unauthorized access and provide efficient solutions for recovery.

There are several major databases available in the market, such as Sybase, Oracle and Microsoft SQL server. In this project, the Oracle database server is applied.

- **Oracle**

The oracle server [24] is an object-relational database management system that provides an open, comprehensive, and integrated approach to information management.

The Oracle database can be defined and manipulated a Structured Query Language(SQL).

- **SQL**

SQL database is a relational database and so the data is simply stored in a set of simple relations. A database can have one or more tables. And each table has columns and rows. Basically, there are two kinds of SQL commands, data definition language (DDL) and data manipulation language(DML). DDL is used to set up the data and DML is used to alter and fetch data (ref to table 4.2).

---

---

<b><u>Structured Query Language</u></b>	
<b>DML</b>	<b>DDL</b>
create database	select
create table	update
create view	delete
drop table	
drop view	

table 4.2

To optimize the SQL query, the data access path is determined at runtime by an optimizer. When a SQL statement is presented to the database, the database engine must first evaluate the statement for correct syntax. The statement is parsed and the parameters in the statement are evaluated. Once the statement has been parsed, the optimizer will determine the most efficient query access paths (ref. to figure 4.2).

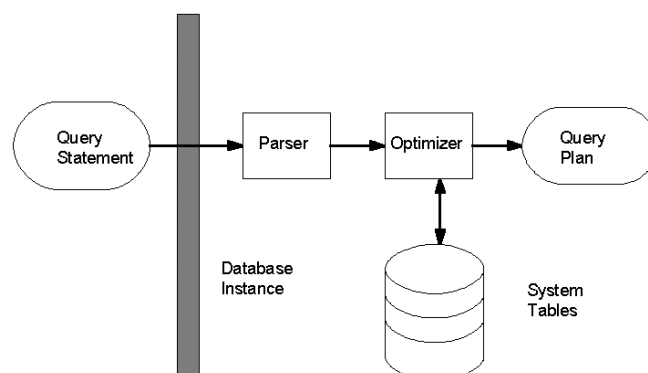


figure 4.2

---

---

## 4.2 Program's Modules

To manipulate a series of operations such as building up the hash table as an index to specific list or removing a specified tuple from a list, it needs to define several modules to achieve these objectives.

### 4.2.1 tuple module:

The structure of a tuple has already been defined in previous section. However, it needs to add two more elements for the sake of the linkage between tuples (ref to figure 4.3). Two added elements are "prev" and "next". The "prev" is used to indicate another tuple that is in front of the current tuple in the list. The "next" is used to indicate the tuple that is behind the current tuple in the list.

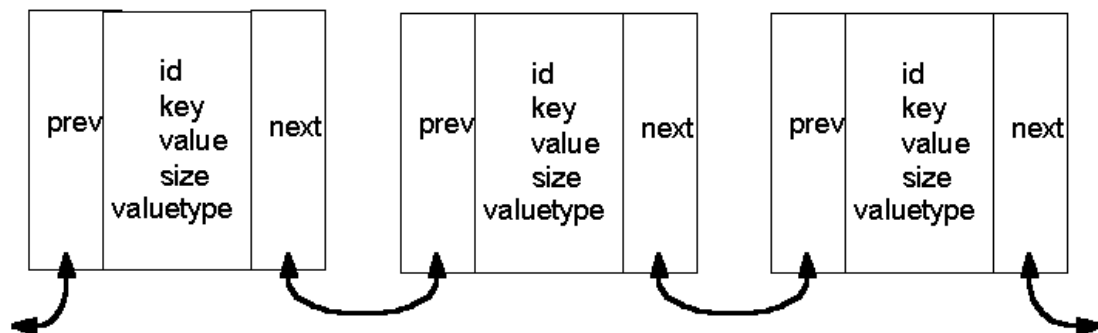


figure 4.3

### 4.2.2 List module:

This module is used to define the structure of an individual list which link up tuples

---

---



assigned by hash function. It includes two elements, "first" and "last" to indicate the head and tail of the list. It also has a element, "length" to show out the length of the list.

#### **4.2.3 ListHandler module:**

This module is responsible for manipulating the operations on a tuple in a specific list. It provides several operations for tuples in a given list. Each operation will return some information to indicate status of manipulation.

- **EmptyList :**

It is used to create an empty list to hold new tuples.

It will not return any values during the operation.

- **AddTuple :**

This operation is used to add a tuple, a "key" and a "value", which is supplied by a user, into a specific list. The mapping of a tuple and a specific list is determined by the hash function based on the "key" submitted by the user and his/her ID. The input parameters are a new tuple to be added and a specific list the tuple will be added to.

It will not return a value but will have some message to indicate the tuple is successful added or not.

- **FindTuple :**

This operation is used to retrieve the existing tuple in a specific list. The list where the seeking tuple expected to be existed in is also compiled by the hash function based on the value of "key" submitted by the user and his/her ID.

Therefore, the input parameters for this operation are the tuple with the submitted "key" and the user's ID and the specific list.

The return value will be the tuple with a "key" matching with the search key or an error message to indicate the failure of searching.

- **FindID:**

This operation is similar to above FindTuple operation but using the user's ID as the search key. It will enumerate all the keys of tuples, which have the matching ID with the search key.

The return value will be sequence of "keys" and an error message will be displayed to a user if no key can be found.

- **RemoveTuple :**

This operation is used to remove the useless tuple from a specific list. The list is also calculated by hash function based on the value of "key" submitted by a user and his/her ID. It will remove all the tuples in the specific list but not only the first matching tuple. In other words, It will return no "value" if the user use the same key for searching again.

The server side will return a message to indicate whether there is a tuple being removed or not.

- **RemoveID**

This operation is used to remove all the tuples with specific "id" from a specific list.

#### 4.2.4 HashTable Module

This module is responsible for the creation of hash table. Its structure is so simple that it only has two elements, an array of "list" and an array holding the numbers from 0 to 255. The operations on them will be defined in the next module.

#### 4.2.5 tableHandler Module

The main mission of this module is to match the input "id" and "key" to the corresponding list by calling the methods, makeT and ourHash, which will be defined in the following section. This module includes the initialization of a new hash table that consists of an array of 256 empty lists. Afterwards, a sequence of operations may be imposed on these lists.

The following operations will be provided:

- **InitialiseList :**

This operation will be executed automatically when the system is woken up. Applying the mechanism of function makeT, 256 empty lists will be initialized and indexed by another array "Tvalue" randomly. The initialized list will be used for whole life period when the server object is available.

- **Insert :**

The input parameters for this operation will be the new tuple with defined values of "key", "value", "size", "valueType" and user's "id" submitted by a user. This operation will call another function ourHash() to calculate an integer between

---

---

0-255 to indicate which list the new tuple should be added to.

- **Find :**

The input parameters for this operation will be a tuple with the search key submitted by a user. it will call the function, ourHash(), to compile an integer between 0-255 based on the values of "key" and "id" of input tuple to indicate which list the seeking tuple will exist in.

- **Delete :**

The input parameters for this operation will be a tuple with the search key submitted by a user. it will call the function, ourHash(), to compile an integer between 0-255 based on the values of "key" and "id" of input tuple to indicate which list the seeking tuple for deletion will exist in.

There are two internal functions called by the operations in this module: makeT() and ourHash().The construction of these two functions is based on the algorithms list at early part of this paper, which describe how to formulate a hashing relationship for the user's id, key and value.

- **ourHash**

Takes an id and a key of string type as arguments and combines them to a single string. Afterwards, submits it into a hash function to return a positive integer in the range of 0-255.

The seed of the table should be static or otherwise a wrong matching will be occurred.

- **makeT**

This function is used to adjust the values stored in the array of Tvalue and make them as if random ordering.

#### **4.2.6 Capability Module**

In order to having access control, a technique of capability is applied in the system. The user should have a password in hand if he/she want to invoke the association service. The capacity, restricted or unrestricted, is generated by server but not selected by a user artificially.

This module is responsible for the generating of capability, which is used to construct a password for a user to login the system of association service and access his/her own associations. As the description in previous section, this module should have the function to generate a new capability for new set of associations and a restricted capability for the accession on already existing set of associations based on the request from its owner.

- **NewCapa:**

According to the ICAP architecture proposed by Li Gong[13], the capability may be generated by a one-way function:

$$\text{Random1} = f(\text{C1}, \text{Object}, \text{Rights}, \text{Random0}).$$

To practice this formation of capability, the system will require the user to key in a user name as the value of C1 explicitly when he/she want to get a new capability

---

---

for the creation of new set of associations. An internal 32-bits hexadecimal number, ID, which is generated by a random function with the current date is used as the value of Object in above function "Random1". Two pre-defined numbers are used to represent the values of full authority and partial authority on accession respectively. The Random0 is a number generated once and for all by random function. Therefore, the function used to generate the capability may be written as the following:

$$\text{new capability} = f(\text{user name, ID, Full Access Right, Random0}).$$

The capability is a 32-bits hexadecimal number. It combines with the ID listed at above to construct a 64-bits password, which will return to the user. The user should submit both the user name and the generated password to login the system to access his/her own associations.

- **RestrictCapa:**

The algorithm to generate a restricted capability is similar to the method that generates a new capability. However, the request on restricted capability should be submitted by the owner of the specific set of associations. For the reason of authentication, he/she must submit his/her user name and the password for checking first. If it is okay, the second part of the password will be abstracted to be the ID. Afterwards, the ID, restricted user name, access right and random0 will be passed into a function to generate a restricted capability. Therefore, the formation of a 32-bits restricted capability can be written as the following:

$$\text{restricted capability} = f(\text{restricted user name, ID, Full Access Right /$$

---

---

Restricted Access Right, Random0).

The restricted capability combines with the ID to construct a 64-bits password, which will be returned to the owner of the set of associations. The owner may pass the restricted user name and restricted password to the authorized user for the accession of the specific set of associations

There are only two kinds of "access right" constructed in this project. However, the fact is that the access right may be classified more detail.

- **Check ID**

This method is responsible for the checking of authentication of the user by the comparison of his/her input user name and password.

It breaks down the input password into two parts. The first 32-bits is the capability and the second 32-bits is the ID of the specific set of associations. The ID, the login user name and the pre-defined constant values of "access right" and Random0 are then used as the parameters of one way function described at section NewCapa and RestrictCapa. If the result is equal to the capability, the checking of authentication is okay. Otherwise, the login action of the user will be refused.

#### **4.2.7 Association Module**

The purpose of this module is to call the functions that use hashing algorithm to store and retrieve associations. It plays a role to integrate the operations of other modules in server side and provides the object implementations called by client program.

---

---

This module will provide the operations for adding, deleting and searching a tuple in the pool of associations:

- **PutAssociation**

This operation is responsible for adding a new tuple in a list in hash table. The input parameters are the "key", "value" and "valueType" submitted by a user explicitly. The user's ID has already been submitted when the user login in the system and so there is no need for the user to submit his/her ID again. the amount of size will be calculated automatically.

It will call the "Insert" operation of module "tableHandler" and the "AddTuple" operation of module "ListHandler" to achieve the target on adding tuple in a list in hash table. It will also call the "outData" operation of the same module to handle the feature of permanent associations.

- **GetAssociation**

This operation is responsible for getting an existing tuple in a list in hash table. The input parameters are the "key", and the user's ID. The user's ID has already been submitted when the user login the system and the value of "key" should be submitted by the user explicitly.

It will call the "Find" operation of module "tableHandler" and the "FindTuple" operation of module "ListHandler" to achieve the target on searching tuple in a list in hash table.



- **DeleteAssociation**

This operation is responsible for the deletion of an existing tuple in a list in hash table. The input parameters are the "key", and the user's ID. The user's ID has already been obtained when the user login the system and the value of "key" should be submitted by the user explicitly. It will also call the "outDeleteData" operation of the same module to handle the feature of permanent associations.

It will call the "Delete" operation of module "tableHandler" and the "RemoveTuple" operation of module "ListHandler" to achieve the target on the deletion of the tuple in a list in hash table.

- **Enumerate:**

This operation is responsible for the enumeration of all the keys of the existing tuples in all the lists in hash table. The input parameters is the user's ID, which has already been submitted when the user login in the system and therefore there is no need for a user to submit any value to invoke this operation. In fact, this operation is implemented implicitly.

It will call the "FindID" operation of module "ListHandler" to achieve the enumeration of keys in a list in hash table with given ID.

Above four operations are used to manipulate the associations those have already loaded in the volatile memory. To achieve the purpose of the operations on the persistent associations and recoverable associations, the following operations need to be added in this module.

- **inData**

This operation is responsible for retrieving all the tuples from a database, which

---

---

is holding the persistent associations. The retrieved tuples will be added into the list of the hashtable based on the values of their "key"s and "id"s. This operation will be executed automatically when the system is initialized.

- **outData**

This operation is responsible for exporting the added tuple into a temporary database for recoverable attribute. It will be called automatically when the "PutAssociation" operation is invoked.

- **outDeleteData**

This operation is responsible for exporting the information of deleted tuples into a temporary database for recoverable attribute. It will be called automatically when the "DeleteAssociation" operation is invoked.

- **recoverData**

This operation is responsible for the recovery of the operations those have not been saved when the server is unpredictably down. The information about the added tuples and deleted tuples will be retrieved into the volatile memory to recover the operations while the system is woken up again.

- **DeleteID**

This operation is responsible for the deletion of the tuples from the volatile memory when the owner of them has decided to leave the system. It will be done on all 256 lists by calling the "RemoveID" method of module "ListHandler" for each list.

- **SaveData**

This operation is responsible for the saving of the operations that have been

---

---

invoked by the user during the client connect with the server. This operation will be invoked when the user wanted to leave the system. The invoked operations may be saved all or none into the database which holding the persistent associations.

#### 4.2.8 Client Module

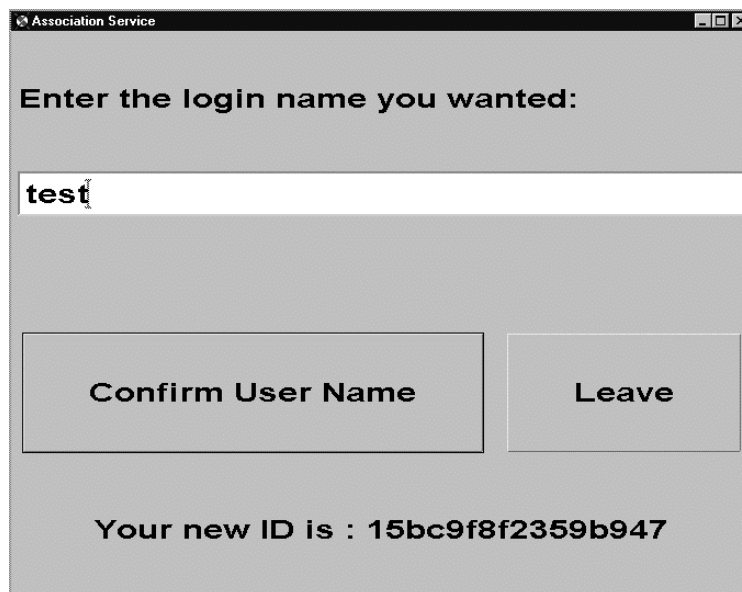
This module is used to define the interfaces for users to access the associations' operations listed in Association Module. Furthermore, it comprises the client side program when the system is in form of client-server based system.

When a user invokes the client program of association service, the following interface will be displayed to the user:



#### 4.2.8.1 New Capacity (Get New ID)

A new user of the association service should get a new password for the login of the system. He/She must press the "Get New ID" to jump into the following interface:



The screenshot shows a window titled "Association Service". Inside the window, the text "Enter the login name you wanted:" is displayed above a text input field containing the word "test". Below the input field are two buttons: "Confirm User Name" and "Leave". At the bottom of the window, the text "Your new ID is : 15bc9f8f2359b947" is displayed.

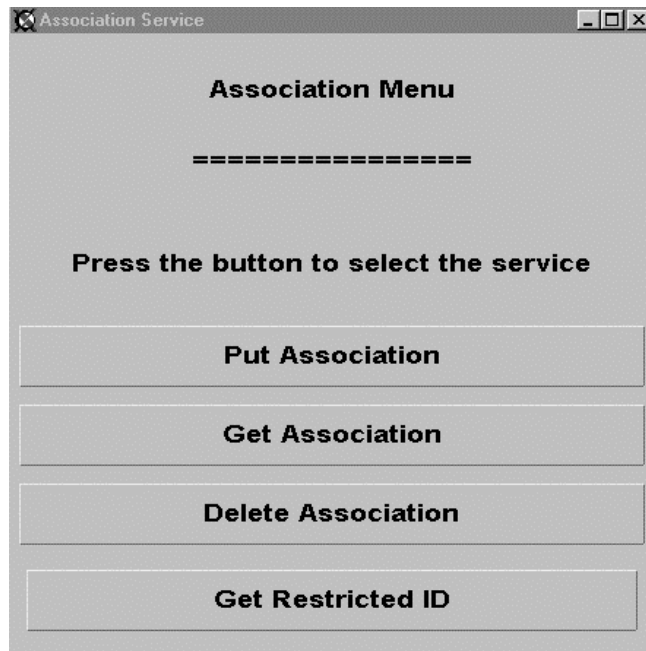
The user is required to key in the wanted login name. Afterwards, the client program will call the "NewCapa" method of "Capability" module to generate a capability and ID to construct a password.

#### 4.2.8.2 Specify "user name" and "password"

This interface is responsible for the access control. The user should input the user name and corresponding password via this interface to login the system.

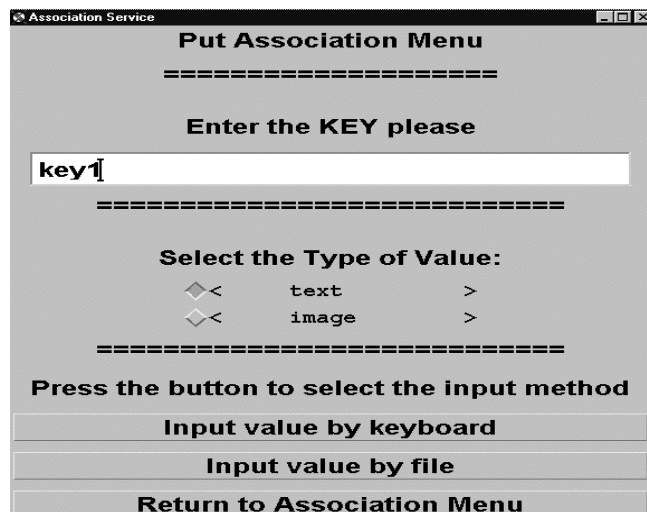
If the user name and password are okay, the user may operate the set of association belonging to him/her. Otherwise, he/she will be blocked at this interface.

When the user has pressed the "confirm" key, the client program will call the "Check ID" method of module "Capability" to check out the authentication of the user. It will jump to the following interface if the checking on authentication can be passed over.



#### 4.2.8.3 Put Association

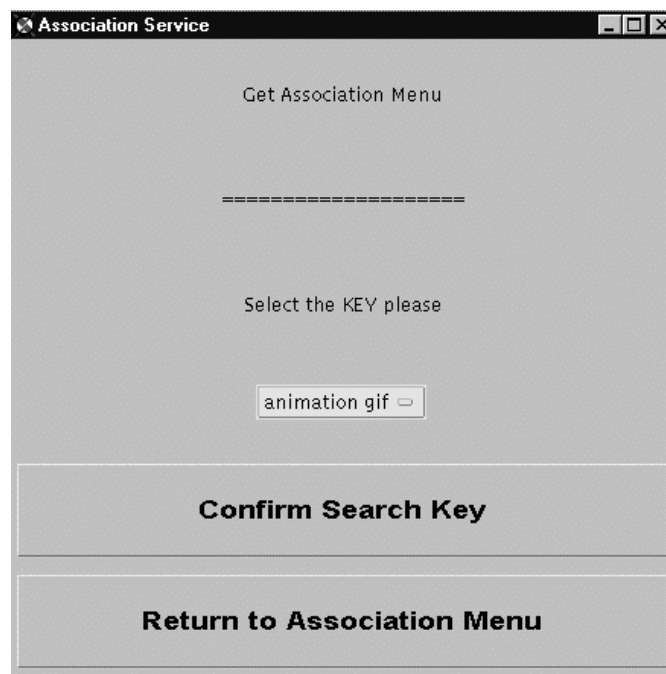
When the button of "Put Association" is pressed, the following interface will be appeared.



This interface is used to prompt the user to submit the values of "key" and "valueType" and select the method for the input of the "value". There are two methods to input "value". One is "Input by keyboard" and another one is "Input value by file". The design of two input methods is based on the consideration of different kinds of data represented by "value". It is impossible to key in the content of "value" if it is a bytes stream of image. When all the data are available, the client program will call the "PutAssociation" method of module "Association" to add the new tuple in the specific list.

#### 4.2.8.4 Get Association

When the button of "Get Association" is pressed, the following interface will be appeared.

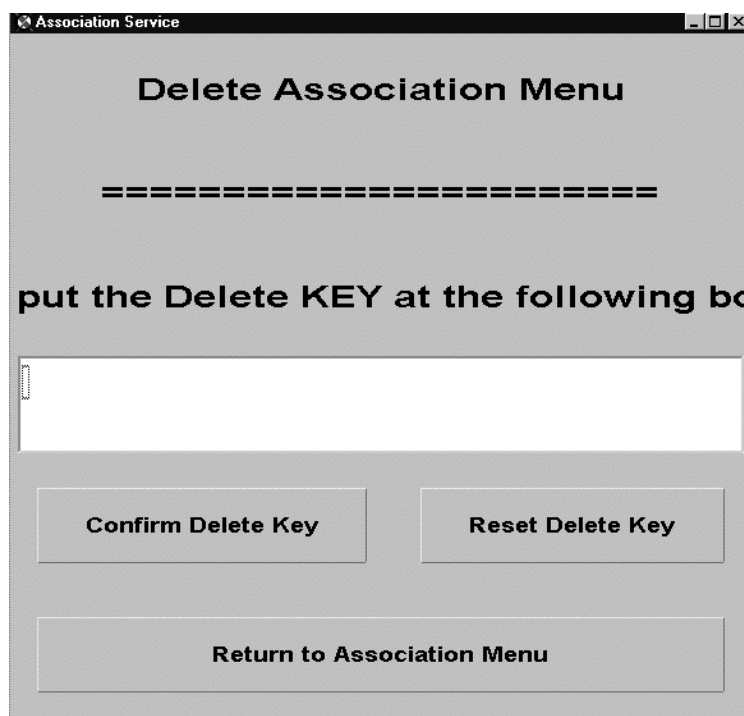


The system will scan the available keys belongs to the user and display them

for the user to select. When the user has selected an appropriate searching key and press the button of "Confirm Search Key", the "GetAssociation" method in "Association" Module will be invoked. The tuple with the matching key will be return if it can be found. In case of the different kind of data represented by the "value", the "valueType" of the tuple will be studied first and then a correct form of "value" can be showed.

#### 4.2.8.5 Delete Association

When the button of "Delete Association" is pressed, the following interface will be appeared.



Through this interface, a user may submit a key to indicate which tuple to be deleted. The submitted key will be passed to the "DeleteAssociation" method of "Association" Module as a parameter. All the tuples with matching keys

will be deleted. A message will be displayed to indicate whether there is a tuple deleted or not.

#### 4.2.8.6 Restricted Capacity (Get Restricted ID)

The button of "Get Restricted ID" is used to invoke an interface for the registered user to generate a restricted password.

This interface is let the user to request the restricted capacity. The user ought to submit his/her user name, password, the restricted user name and the access right expected to authorize. After the user has entered every required information and presses the "confirm" key, the system will call the "restrictCapa" method of "Capability" module to process the generating of restricted capability. The "restrictCapa" will call another method, "CheckID" for the checking on the authentication. If the checking is okay, the "restrictCapa" method will use the authorized access right and the restricted user name as parameters to generate a password and return to client side.

The screenshot shows a dialog box titled "Association Service" with the following content:

Fill the following box to retrieve the Restricted ID

Current Login Name : test

Current User ID : 19d9b8f4768fd95

Restricted User ID : sam

Access Right assigned :

- < Operate All Function >
- < Read Funtion only >

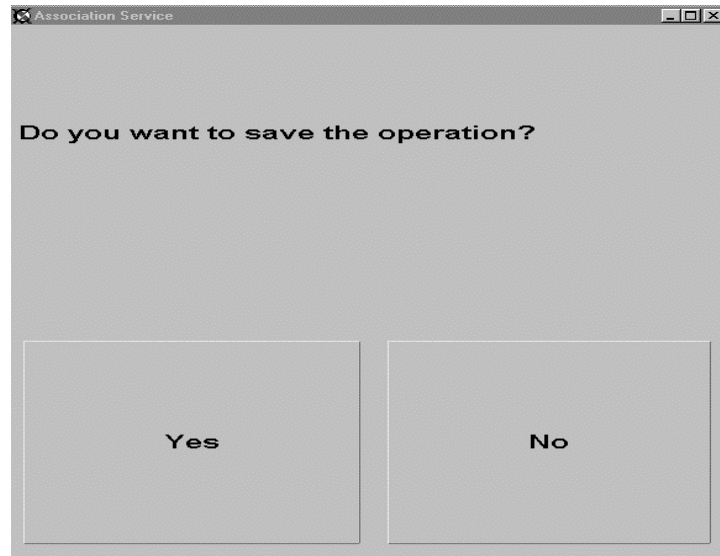
Confirm Leave

The restricted ID of sam is 5648cef14768fd95



#### 4.2.8.7 Save the operations on associations

When a user wants to stop the accession on the system, the following interface will be invoked automatically to prompt him/her to determine whether the modification on associations is saved or not.



If the selection is "Yes", all the modification including the "put association" and "delete association" will be used to update the database. Meanwhile, all the associations belong to this user will also be removed from the volatile memory.

### 4.3 System Development

The project has been broken to a few phases in order to develop the application systematically.

### 4.3.1 Phase I: a single program

Firstly, a single program is constructed running in a single machine, PC or Unix platform, to experience the manipulation of the association service. Although it is just a program in the stage of prototype construction, it still includes the following modules:

- Client Module
- Association Module
- tableHandler Module
- hashTable Module
- ListHandler Module
- List Module
- Capability Module

The functions provided by above modules in the single program are similar to those described in section 3.2, but some operations of the modules is disable. For example the operations for input or output the data from or to the database have not been included.

Accordingly, all the operated associations are stored only in the volatile memory and could not be recoverable when the program is restarted from the sudden crash of the machine or normal exit procedure. All associations ought to be re-entered for manipulation. Because it is only running in a single machine, it does not have the features of client server based system. However most of the user interfaces

---

---

have already been defined in this program.

The user should specify his/her login user name and corresponding password first if he/she wants to request the association service. If not, he/she may get a new password by invoking the operation of "new capability".

If a user may successful login the system, he/she may operate the functions of adding, getting or deleting the associations.

#### **4.3.2 Phase II : Construct a program in Client-Server based approach**

The construction of the programs in this phase is based on the structure of program in phase I. The program has been split to a client program and server program. Both of them are expected to be able to run on the different machine. It has employed a middleware, CORBA for the server location transparency by using CORBA's naming service. Based on CORBA, the components of the system may communicate with each other once the binding between the ORB and them has been occurred.

The Visigenic's VisiBroker has been selected as ORB for CORBA implementation. The Client program and Server program can communicate via the ORB as if they are in the same machine.

As the previous section said, it should define the IDL first for CORBA implementation.

#### 4.3.2.1 Step 1) Define IDL

In order to deal with the occurrence of error, a few user-defined exceptions are defined in the IDL:

- 1) An error will be occurred if the tuple cannot be found during the searching of key from a list, and so an exception needs to be defined:

```
exception tupleNotFoundException{ };
```

- 2) An error will be occurred if the "id" cannot be found during the searching of "id" from a list, and so an exception needs to be defined:

```
exception idNotFoundException{ };
```

- 3) An error will also be occurred if the password doesn't match the login user name, and so an exception needs to be defined:

```
exception idNotMatchException{ };
```

Based on the modules defined in section 4.2, the corresponding interfaces are defined by IDL as the following:

##### 1) **tuple module:**

```
typedef sequence<octet> ByteArray;
```

```
interface tuple {
```

```
    attribute long long id;
```

```
    attribute string key;
```

```
    attribute ByteArray value;
```

```
    attribute long size;
```

---

---

```
attribute string valueType;  
  
attribute tuple prev;  
  
attribute tuple next;  
  
};
```

## 2) List module

```
interface List {  
  
    attribute long length;  
  
    attribute tuple first;  
  
    attribute tuple last;  
  
};
```

## 3) ListHandler module

```
interface ListHandler {  
  
    List EmptyList();  
  
    void AddTuple(in tuple addtuple, in List givenlist);  
  
    void RemoveTuple(in tuple removetuple, in List givenlist)  
  
        raises(tupleNotFoundException) ;  
  
    tuple FindTuple(in tuple findTuple, in List givenlist)  
  
        raises(tupleNotFoundException) ;  
  
    string FindID(in long long id, in List givenlist)  
  
        raises(idNotFoundException) ;  
  
};
```

## 4) HashTable Module

```
typedef List ListArray[256];
```

---

---

```
typedef long TArray[256];  
  
interface hashTable {  
  
    attribute ListArray listArray1;  
  
    attribute TArray Tvalue;  
  
};
```

### 5) tableHandler Module

```
interface tableHandler {  
  
    hashTable InitialiseList();  
  
    List Insert(in tuple InsertTuple, in hashTable table);  
  
    List Find(in tuple Findtuple, in hashTable table);  
  
    List Delete(in tuple DeleteTuple, in hashTable table);  
  
};
```

### 6) Association Module

```
interface Association {  
  
    void PutAssociation(in long long id, in string key, in ByteArray value, in long  
size, in string valueType);  
  
    tuple GetAssociation(in long long id, in string key)  
        raises(tupleNotFoundException) ;  
  
    void DeleteAssociation(in long long id, in string key)  
        raises(tupleNotFoundException) ;  
  
    string Enumerate(in long long id)  
        raises(idNotFoundException) ;  
  
    void SaveData(in string saveD);
```

---

```
void RecoverData();  
  
void inData(in string dbfName);  
  
void DeleteID(in long long id);  
  
};
```

### 7) Capability Module

```
interface Capability {  
  
    string newCapa(in string useName);  
  
    string restrictCapa(in string userName, in string userID, in string  
newUserName, in string right)  
  
        raises(idNotMatchException);  
  
    string checkID(in string userName, in string userID)  
  
        raises(idNotMatchException);  
  
};  
  
};
```

#### 4.3.2.2 Step 2) Implement these interfaces with Java class

The second step is to use an IDL compiler to compile the IDL. IDL itself is independent from the programming language. What code to be generated is dependent on the kind of IDL compiler to be used. It is planned to use Java to develop the system. In case of that, the visibroker's Java IDL compiler is used to generate the stub codes and skeleton codes.

After the compilation of IDL file by calling the command: "idl2java", a series of

---

---

Java files are generated. There are several files generated for each interface defined in the IDL file. For example, there are `_st_Association.java`, `AssociationHelper.java`, `_AssociationImplBase.java`, `Association.java`, `_example_Association.java`, `AssociationHolder.java` and `AssociationOperations.java`, are generated for the interface `Association`. `_st_Association.java`, `AssociationHelper.java` and `AssociationHolder.java` are classified as the Client-side Java and the others are classified as the Server-side Java.

Each of them has their own function.

- **`_st_Association`**

It is a Java class that implements the client-side stub for the `Association` object.

It is an internal implementation of the `Association` interface that provides marshaling functions. It should be included in the client implementation.

- **`_AssociationImplBase`**

It is a Java class that implements the CORBA server-side skeleton for `Association`. It unmarshals the arguments for the `Association` object. It brings together the CORBA and Java object models. It does this by being a Java object that also implements the Java `org.omg.CORBA.Object` interface, which is the root CORBA interface all CORBA objects must implement. For the object implementation, a Java program should be constructed with the inheritance from this pre-defined Java class.

---

---



- **AssociationHelper**

It is a Java class that provides useful helper functions such as narrow function and bind function for the client program. The clients may use it to locate objects of this type via the ORB.

- **AssociationHolder**

It is a Java class that holds a public instance member of type Association. Through Association Holder, clients and servers pass objects of type Association as "out" and "inout" parameters inside method invocations.

- **Association**

It is a Java interface mapped from the interface defined in IDL file. It is only an interface and therefore it needs to have extra code defined for implementation.

- **\_example\_Association**

It is an example class for the Association object implementation. It contains constructors and example methods such as PutAssociation, GetAssociaton and so on those were defined in IDL file. The methods have already defined in phase I. Therefore, the Association object implementation is easy to be completed by filling the methods in the example.

#### 4.3.2.3 Step 3) Object Implementations

Each interface should have a Java file defined for object implementation. Accordingly, `tupleImpl.java`, `ListImpl.java`, `ListHandlerImpl.java`, `HashTableImpl.java`, `tableHandlerImpl.java`, `AssociationImpl.java` and `CapabilityImpl.java` are constructed.

It needs to create a server program for the initialization of the ORB environment and starts objects. This sever program has the main method to:

- 1) initialize the ORB,
- 2) initialize the BOA,
- 3) create a `AssociationImpl` object and `CapabilityImpl` object,
- 4) export the newly created objects to the ORB,
- 5) wait for incoming requests from client side.

#### 4.3.2.4 Step 4) Creation of Client program

The user interfaces are defined in client program. Besides, the client program has the main method to perform the following functions:

- 1) initialize the ORB,
- 2) locate the remote `Association` object and `Capability` object,
- 3) submit the request and receive the reply.

#### **4.3.2.5 Step 5) Run the Client/Server Program**

To run the Client and Server program, it should start the Visibroker ORB Smart Agent (osagent) first.

The osagent is a dynamic, distributed directory service that provides facilities for both client applications and object implementations. When a client application invokes the bind method on an object, the osagent locates the specified implementation and object so that a connection can be established between the client and the implementation. Object implementations register their objects with the osagent so that client applications can locate and use those objects. When an object or implementation is destroyed, the osagent removes them from its list of available objects.

After the byte codes of Client and Object Implementations have already been generated, the Client and Server can be run by the command: vbj, which is used to start up the local Java interpreter.

The Client and Server can be woken up in different machines and the server is transparent in location. The Client and Server are connected via the ORB indirectly. All the operations of the association service is the same as those of the single program defined in Phase I.

The program developed in this phase still cannot handle the permanent and recoverable associations. These attributes are added in Phase III.

### 4.3.3 Phase III: Permanent and Recoverable Attributes of the Associations

In order to make the associations created to be persistent and recoverable. The Oracle database is applied in the project to store the associations (ref to figure 4.4).

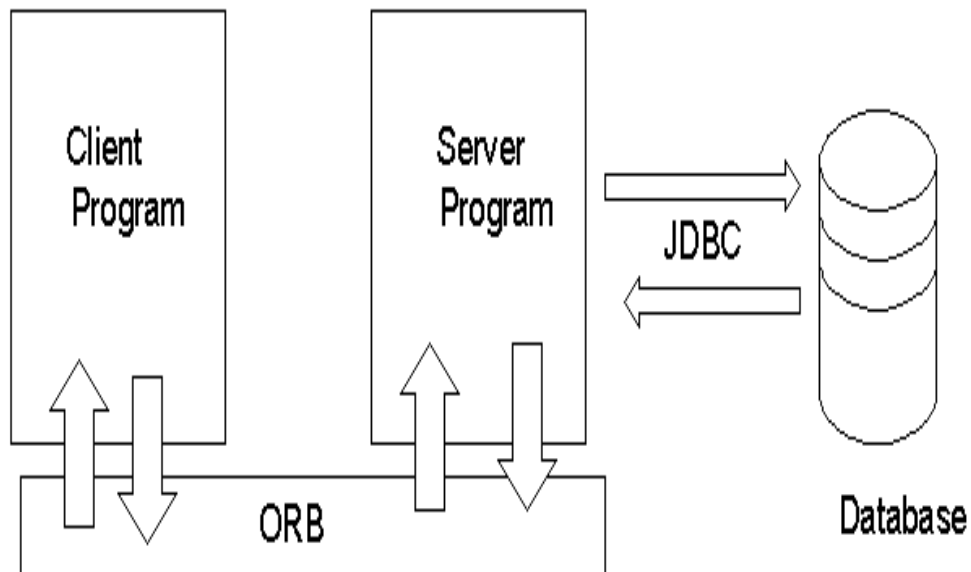


figure 4.4

Two tables of the database are created as "association" and "tempasso" for the storage of associations. The table "association" is used to hold the permanent associations and table "tempasso" is a temporary table created for recoverable attribute.

If the system is being closed in a normal procedure, the user will be asked whether the modification on associations is saved or not. If the user confirms to save the modification, the information stored in table "tempasso" will be read and used to update the permanent table "association". Afterwards, the table "tempasso" will be emptied.

However, if a sudden crash of the association service system is occurred, there is no way for the user to save the modified associations into the permanent table: "association" for future retrieval. However, the damaged manipulation on association can be recovered into the volatile memory because of the existence of temporary table, "tempasso". The table "tempasso" has held all the information about adding or deleting associations during the system operated.

Every time a user can login the system successfully, the associations belonging to this user will be read from both table "tempasso" and the permanent table "association". The table "tempasso" is read after the retrieval of data from table "association" has completed.

If there is no sudden crash of the system , the table "tempasso" will be empty and no information about the modification of associations can be read from it. Otherwise, the information for recovery is read and used to update the associations stored in the volatile memory. Based on this mechanism, the associations can have the persistent and recoverable attributes.

In order to distinguish the kinds of manipulation on data, besides the columns created for holding the elementary variables of a tuple, one more column, "flag1", is created to hold a flag that indicates the kind of modification on association. If the value of flag1 is "delete", the corresponding association will be removed from the system. Otherwise, the added or updated association will be retrieved into the volatile memory.

To hold the related values, the database is designed as the following:

---

---

**"association"**

<u>Name of Column</u>	<u>Oracle Datatypes</u>
id1	number(20)
key1	varchar2(256)
value1	long raw
size1	number(10)
valuetype1	varchar2(256)

table 4.3

**"tempasso"**

<u>Name of Column</u>	<u>Oracle Datatypes</u>
id1	number(20)
key1	varchar2(256)
value1	long raw
size1	number(10)
valuetype1	varchar2(256)
flag1	varchar2(10)

table 4.4

**Summary of datatypes used:**

varchar2(size) : Variable-length character data. A maximum size is specified in the parenthesis.

number(p, s) : Variable-length numeric data. "p" is precision and "s" is scale. if no precision is specified, the column stores values as given. If a scale is not specified, the scale is zero.

long raw :            Variable-length raw binary data.

The second middleware, JDBC, is applied for the connection between the Java program and the Oracle database. The JDBC has provided the means to query and maintain a distributed database.

#### **4.3.3.1 Steps to involve the database via JDBC program:**

##### **1) Load driver**

The first step in using JDBC is to load the JDBC-ODBC bridge driver to create an environment in the program such that the following step can be done. The "forName" method of "Class" object is used to achieve this target.

##### **2) Create connection**

It needs to establish a connection to the database before any database-specific SQL statements in the Java program. The connection is established by the calling to the "DriverManager getConnection" method.

##### **3) Create statement**

After the establishments of the driver and the connection, It needs to define a "Statement" object to manage the SQL statements. The "createStatement" of "Connection" class is used to accomplish this creation. The alternative of "Statement" object is "PreparedStatement" object which allows the SQL statement to be sent to the database for parsing and optimization before execution.

##### **4) Execute statement**

---

---

After the creation of "Statement" object or "PreparedStatement" object, The SQL statement can be executed by calling "executeQuery" or "executeUpdate". General speaking, the "executeQuery" is applied for the query to the database whereas the "executeUpdate" is used when there is modification on the database.

**5) Iterate ResultSet if there is some value returned from database.**

The "ResultSet" is a JDBC Java class for the collection of results retrieved from the database by query statement. In case the retrieval of results is row by row, the "ResultSet" provides a method for iteration.

Based on JDBC, the server program may save the new or updated associations in the database for permanent attribute. The editing associations can also be recoverable even the sudden crash of the system.



## 5. System Requirement

The association service is designed in client-server based approach with CORBA implementation, and so it is expected to run on the distributed environment. All the programming codes for the system is a Java codes. Java is a programming language designed especially for the use in internet and one of the strong point is machine independent. Therefore, theoretical speaking it may run on both PC and Unix workstation if the required software exists.

### Software requirement

- JavaSoft JDK 1.1 version or above
- Visigenic VisiBroker for Java
- Oracle database
- X server for MS windows if display on PC

## **6. Experience for the Development of the Project**

There are so many constraints and consideration during the construction of this system. It needs to select which programming language to be utilized and which method for the retrieval of data. It also needs to consider the availability of hardware and software. The following description is some experience obtained during the development of this system.

### **6.1 Usage of programming language**

Originally, it is planned to use C++ for coding, but it has changed to use Java to develop the system because there are abundance of libraries be able to be called. Consequently, the Java ORB visibroker is applied for the CORBA implementation.

Although IDL is not an actual programming language and it is only a design on the interface, coding on the server program and client program becomes more easily because of the utilization of IDL.

### **6.2 Application and applet**

If it implements a client as a Java application, there is no need to worry about the restrictions that exist for applets for the security control. It is simply running the client and server programs in their Java virtual machine (JVM) and using the ORB for interaction. However, the client codes is running in remote Unix machine and it needs an X server for MS windows for the export of display from the Unix application.

On the contrary, if a client is developed as an applet, the applet byte code of the client is exported to the local machine and running in the JVM included in web browsers

---

---

such as Netscape, Internet Explorer. There is no need for a user to install an extra X server for MS window if he/she is using PC as local machine. It may browse the character of Unicode easily if the web browser supports.

Java applet is designed to implement in the internet environment and the security issue is much concerned. In case of that, applet sandboxing is responsible to introduce the limitation of the accession on the devices of the local machine to prevent the unpredictable destruction done by an untrusted applet. It allows applets having the connection to only the host where they were downloaded from. An applet client is restricted to invoke only the operations of objects that are local or reside on its host of origin.

For CORBA, however, one of its missions is to provide location transparency. It means that a client can invoke operations on the object but regardless of their physical location. Obviously, the existence of Java applet sandboxing is in conflict with this attribute of CORBA.

In case of that, the "CORBA Implementation on Association Service" is developed as a Java application. In fact, there is a way to solve the problem of conflict and this will be discussed in Chapter 7.

### **6.3 Data retrieval from database**

In order to having persistent and recoverable association service, it needs to have a connection of the Server program with the database and there are the input and output operations on associations between the volatile memory and the secondary storage. There may be different arrangements on the data flowing between memory and

---

---

database:

- **Approach 1**

One of approaches is to retrieve all the saved associations into the memory when the client side program is woken up. All the operations on the association such as adding association, getting association and deleting association will be done in the volatile memory. It will only have a backup about the operation being done in a temporary table of the database for recoverable. And therefore, it should handle the data structure for searching and grouping. The tuple, list, hash table and their handler are constructed for these objectives. All the manipulations done on the associations will be saved to a permanent table in the database only when the client leaves the system.

- **Approach 2**

Another approach is to interact with the database each time when there is a manipulation done on the associations. When there is a new association input from client side, it will be saved to the permanent table of the database directly. When there is a request on searching an existing association, it will search the database directly. In case of that, there is no need to design extra modules to speed up the searching. It is because the data in the database may be search in sequential or by index.

In first approach, most of the manipulation of data is done in the volatile memory. It is known that running in the memory is much faster than in the hard drive. Therefore, it will be more efficient for searching in first approach compared with second one.

General speaking, however, the volume of memory is much less than that of

---

---

secondary storage. If the number of associations is larger and larger, the memory will be used up and out of space for adding associations. The situation will be reversed in second approach.

In this project, it has selected the first approach because there is a little amount of sample associations. However, if it is a project for business, which is expected to handle a large amount of data, it is better to use second approach or combine them for implementation.

## **6.4 Index on database**

An index is a database object that can speed up the retrieval of rows from the table. Its purpose is to reduce the input and output of disk by using a B\* Tree indexed path to speed up the searching of data. The index is automatically used and maintained by the Oracle Server. There is no direct activity required by the user when an index has been created.

Each index is composed of column values that the index is on any pointers to the row containing that value. The pointer points to the row in the table directly, and so there is no need for a full table scan.

As the previous paragraph said, index is also a database object. The creation of index will occupy disk space in the form of index page space and distribution pages. Furthermore, More indexes on a table do not mean that it will speed up queries. It is because the DML operations such as "update", "insert" and "delete", on a table with indexes will lead to the updating of indexes, which will slow down the queries.

---

---

Therefore, it is not necessary to create index. The following factors should be consider before willing to create the index:

- 1) The size of table;
- 2) The frequency of "where" clause will be used;
- 3) Percentage of existing rows will be queried;
- 4) The frequency of the table will be updated;

If the table is small and it is always updated, there is no need to create an index. The table created for this project will hold only a small amount of sample associations and the table will be updated frequently, and therefore no index is created on the tables of this project.

## **6.5 Limitation of the system**

This system is designed in client-server based approach. It is expected to be a multi-user system. In case of that, each user has his/her own login password and can access only his/her set of associations. It has also been designed that the owner of a specific set of association may assign a restricted access authority to another one. In this system, however, two or above users with the right to access a same set of association can not manipulate the associations at the same time. It is a draw back of this system.

Besides, theoretical speaking, the system may display 2-bytes character such as the Chinese character, but it depends on the supporting of the operating system where the client exports the display to. The system itself does not have a module to handle the display of Chinese character.

Although this system is machine independent, it is always running in Unix workstation because of the requirements on installed software such as Visibroker Java ORB and Oracle database.

---

---

## 7. Further Development

This system is not fully developed. In fact, there may be further development based on this system.

### 7.1 Develop client side of the system as an applet

Although it has the conflict between the CORBA implementation and the restriction from applet sandboxing and firewall with the CORBA implementation, there are ways to solve this problem. One solution is to put a forwarder, IIOP (Internet Inter-ORB Protocol) gateway, on the web server (ref to figure 4.5).

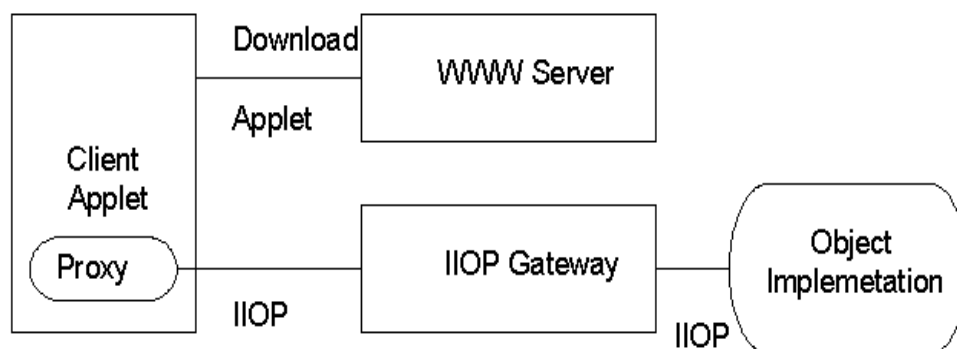


figure 4.5

The forwarder program lets the CORBA-enabled applets loaded from web server invoke operations on the IIOP gateway instead of an actual object.

The next candidate is let the applet be signed by an identity marked as trusted in the identity database. The signed applet indicates that the applet can directly connect to servers if users agree to grant the applet the requested privileges.

If the coding for the client side of the system may be converted to be an applet, users may directly access the association service by the web browser and need not wake up another application software.



## **7.2 Adding more interface for various format of "value"**

In current system, it may display the "value" as a text or an image. In fact the "value" may represent another format of data, such as sound clips or video frames. The JDK 1.1 version, which is used to develop this system, does not support the playing of audio clip in Java application. However, the audio clip can be played if the system is designed as an applet.

## **7.3 Multi-users supporting**

It is possible that more than one user are willing to access the same set of associations at the same time. To enable the system having this function, the concurrent control of the I/O of data should be imposed to avoid the tricks for data sharing such as dirty read and premature writes.

## **7.4 Directory service as a yellow page**

The association service designed in current developed system is to reply only a single value for a key submitted. It is one to one mapping. This feature may be advanced to be that a set of values is returned when a single key is submitted. The mapping in this manner is known as the directory service.

The directory service is as same s the function of yellow page. Usually, a user would like to know a group of information but not only a single data. When a user keys in a word, "restaurant" for example, he/she expects to get a group of names and their

address but not only a single "value". Therefore, the directory service may be a potential project title.

## 8. Conclusion

Association Service is the service for looking up a value by submitting a key. Its concept and structure is so simple that it is easy to port to other application, especially when there is rapid development of internet. For example, the business of a very successful company, Yahoo, is to provide a searching service in the internet. Information searching in the internet is a very hot business currently. Searching service can be classified as a kind of association service running in internet, a very big distributed system.

To construct an application in a distributed system is a challenge to the programmer. With CORBA, A programmer develops an application in distributed system becomes much easier than that without this middleware. Its IDL is easy to learn and helpful. However, the DCOM is its potential competitor and it most likely that DCOM will be more popular in future because of the aggressive of its vendor, Microsoft.

## Reference

- [1] Osman-Allu, N.A. Telecommunication Interfaces for Deaf People. Social Needs and the Interface, IEEE Colloquium on, Pages: 811-814.
- [2] Chris Hanson, the MIT Scheme Team and a Cast of Thousands. MIT Scheme Reference. Edition 1.62 for Scheme Release 7.4. 16 April 1996.
- [3] Lawrence S. Orilia. Computers and Information, An Introduction. McGraw-Hill International Editions, 1986.
- [4] John A. Stankovic. A perspective on Distributed Computer Systems. IEEE Transactions on Computers, Vol. C-33, No. 12. December 1984.
- [5] James N. Gray. An Approach to Decentralized Computer Systems. IEEE Transactions on Software Engineering, Vol. SE-12, No. 6, June 1986.
- [6] Sean Baker. CORBA Distributed Objects, Using Orbix. Addison-Wesley. 1997.
- [7] Andrew D. Birrell and Bruce Jay Nelson. Implementing Remote Procedure Calls. ACM Transactions on computer Systems, Vol. 2, No. 1, pp. 39-59, Feb 1984.
- [8] OMG. The Common Object Request Broker: Architecture and Specification, 2.0 Edition, July 1995.
- [9] Eila Niemela, Mikko Holappa. Experiences with the Use of CORBA, Euromicro Conference, 1998. Proceedings. 24th Volume: 2 , Page(s): 989 -996 vol.2
- [10] Robert Orfali, Dan Harkey. Client/Server Programming with JAVA and CORBA. 2<sup>nd</sup> Edition. 1998
- [11] Instructor's Guide from Coulouris, Dollimore and Kindberg. Distributed Systems: Concepts and Design. Edition 2. Addison-Wesley Publishers 1994.

- [12] I-Lung Kao, Chow R. An extended Capability Architecture to enforce dynamic access control. Computer Security Applications Conference, 1996., 12th Annual , Page(s): 148 -157
- [13] Li Gong. A Secure Identity-Based Capability System. Security and Privacy, 1989. Proceedings., 1989 IEEE Symposium on , Page(s): 56 -63
- [14] Peter Pearson. Fast Hashing of Variable-Length Text Strings, Communications of the ACM, Vol. 33, No. 6, pp 677-680 (June 1990)
- [15] John Rosenberh, Alan dearle, David Hulse, Anders Lindstrom, and Stephen Norris. Operating System Support for Persistant and Recoverable Computations.
- [16] Atkinson, M.P., Chisholm, K.J., and Cockshott, W.P. PS-Algol. An Algol with a persistent heap. ACM SIGPLAN Notices 17, 7 (July 1981), 24-31.
- [17] Atkinson, M.P., Bailey., Chisholm, K.J., Cokshott, W.P., and Morrison, R. An approach to persistent programming. Comput. J. 26, 4 (1983), Page(s) 360-365.
- [18] 4.3 Berkeley Software Distribution, UNIX User's Reference Manual, USENIX Association, Berkeley, California, April 1986.
- [19] Eliezer Levy and Abraham Silberschatz. Distributed File Systems: Concepts and Examples, ACM Computing Surveys, Vol. 22, No. 4, December 1990.
- [20] Dean Thompson and Damien Watkins: Comparisons between CORBA and DCOM : Architectures for Distributed Computing. Technology of Object-Oriented languages, 1997. TOOLS 24. Proceedings, 1998. Page(s) 278-283.
- [21] Andreas Vogel and Keith Duddy: Java Programming with CORBA, 2<sup>nd</sup> Edition, Wiley Computer Publishing.
- [22] Barry, D.; Stanienda, T.: Solving the Java Object Storage Problem, Computer,

Volume: 31 11, Nov 1998, Page(s): 33-40.

[23] Art Taylor: JDBC Developer's Resource, Informix Press, 1997

[24] Lefty Leverenz: Oracle8 Concepts, Release 8.0 Oracle Corporation

---

---

## Appendix I - System Installation Guide

- 1) Register first (go to Rm903) or send email to Corba administrator with "VisiBroker registration" in the mailsubject.(Without registration, you are unable to run visibroker successfully.)
- 2) Read /usr/local/vbrokerj/README before using VisiBroker for Java.
- 3) To setup the environment for VisiBroker to work properly, you have to
  - a. "/usr/local/vbrokerj/misc/startup -f" to reinitialize everything.
  - b. "source ~/.vbrj/vbroker.csh" if you are using c-style shell, or  
". ~/.vbrj/vbroker.sh" if you are using sh-style shell
  - c. "setenv PATH \${PATH}:/usr/local/JDK/bin" c-stye shell  
"PATH=\${PATH}:/usr/local/JDK/bin;export PATH" sh-style shell
- 4) "make"
- 5) To setup the environment for Oracle 8 database server (CSE user only), you have to  
"source /usr/local/oracle8/setup".
- 6) To setup the environment for using the JDBC with Oracle and OCI driver, you have to  
"setenv CLASSPATH :.\$ORACLE\_HOME/jdbc/lib/classes111.zip:\${CLASSPATH}".
- 7) "osagent &"

If it is unable to bind to default port, you may change the  
OSAGENT\_PORT by key in "setenv OSAGENT\_PORT <new port number>  
For example: setenv OSAGENT\_PORT 26262 and try to bind osagent again.
- 8) "vbj Server" (start the server, and wait for the message that indicates the server

being ready)

9) Assuming your local machine is <hostname\_a>, and you want to run the client in

<hostname\_b>, then

a. In <hostname\_a>, do "xhost + <hostname\_b>" to allow the external host.

b. In the window for <hostname\_b>, do

```
"setenv DISPLAY <hostname_b>:0.0"          csh-style shell
```

```
"DISPLAY=<hostname_b>:0.0;export DISPLAY" sh-style shell
```

(If you are running the client at the local machine, then you don't have to do these

step a and b.)

c. Run the client "vbj Client" in the <hostname\_b> window

Note: 1) If your local machine is a PC with MS windows as an OS, you must install a

X server for MS windows for the display.

2) You may refer to the user\_guide.txt for the system operation.



## Appendix II - User Guide

When you has woken up the Client program based on the instructions of "system installation guide", a "Main Menu" will be invoked" first.

The following content describes the operations of the system:

### 1) Main Menu

#### 1.1) Get New Capability

It is used to get a new password to enter the system.

You need to enter the wanted login name and then a password will be generated for you.

#### 1.2) Login the system

You are required to enter the login name and password

### 2) Association Menu

#### 2.1) Put Association

It is used to add a new association in the system.

When you choose this operation, you will be prompted to enter the "key", "value", the format of "value" step by step.

There are two formats: text or image.

#### 2.2) Get Association

It is used to retrieve the association from the system.

When you choose this operation, you will be prompted to enter a searching

key to get the related data.

### 2.3) Delete Association

It is used to delete existing association.

When you choose this operation, you will be prompted to enter the searching key to delete all the related associations.

### 2.4) Get Restricted Capability

It is used to generate a restricted password to other one whom you wanted to authorize the access right.

There are two type of access right: all or "get association" only.

## 3) Save Modification

When you close the window to leave the system, you will be asked to indicate whether the modification on associations is saved or not.

If you select "Yes", all the modification done will be saved to permanent situation.