BROWN UNIVERSITY
Department of Computer Science
Master's Thesis
CS-92-M17

"Integrity Constraints for Object-Oriented Database System"

by

Yu-Fang Li

Brown University
Department of Computer Science
Master's Project

"Integrity Constraints for Object-Oriented Database System"

by

Yu-Fang Li

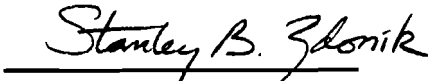
Master's Project

Integrity Constraints for Object-Oriented Database System

Yu-Fang Li

June 30, 1992

Submitted in partial fulfillment of the requirements of the degree of Master of Science in the Department of Computer Science of Brown University



Stanley B. Zdonik
advisor

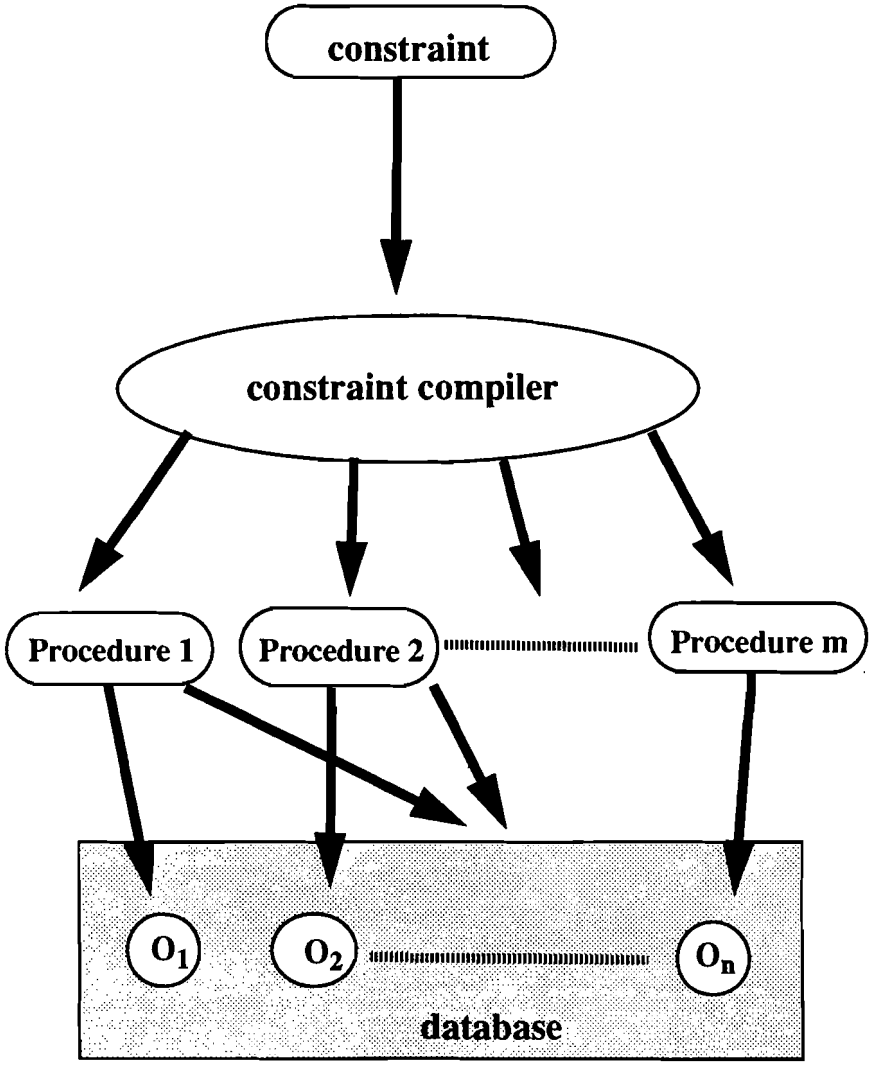
Content

Chapter 1: Introduction

Objective:

In this project, we built a system on top of the ObjectStore OODB to enforce integrity constraints. The system can parse an input constraint and generate a set of ObjectStore compatible C++ procedures automatically. Then, it will attach the generated procedures to those objects that would be affected by this constraint and initialize a constraint instance to store all the checked values. Every time when these objects are updated, the system will check these constraints automatically by executing those generated procedures to ensure data consistency and integrity.

Flowchart:



Chapter 2: Implementation

In this system, first it parses the input constraint, splits the classes along the paths in the constraint into set AD and set AP. Then, the system will generate a set of procedures for those classes in the path according to which set (AD or AP) these classes belong to. Finally, the system will attach these procedures into the objects in the database.

This section describes how to generate the demon procedures and attach these procedures to the relative objects. There are 2 parts of it: the first part talks about the format of constraint language, the algorithm to generate demons and how to declare and initialize the constraint instance. The second part talks about how to attach the procedures to those objects that will be affected by the constraint and some difficulties in ObjectStore which unfortunately resulted in requiring users to write some functions on their own.

2.1. Demon Generator

This section is mainly focused on the implementation of demon generator. First we'll see the grammar for the constraint languages in section 2.1.1. Then, section 2.1.2 has a full description of the demon generating algorithm. Section 2.1.3 describes what kind of constraint instances we can have and how to decide which type should be declared in the generated code. After presenting the strategy of demon generator in this section, section 2.3 will talk about the implementation about attaching the procedures and some difficulties in ObjectStore which result in limitations of this system.

2.1.1 constraint representation

The language for constraint is evolved from the attribute grammar for C++. The grammar for the constraint is as follows:

program: prog_decls
;
prog_decls: declaration
| prog_decls declaration
;
declaration: opt_decl_specifiers constraints
;
opt_decl_specifiers:
| opt_decl_specifier
| opt_decl_specifiers opt_decl_specifier
;
opt_decl_specifier: class_specifier
;
class_specifier: class_head '{' opt_member_list '}'
;
class_head: CLASS decl_identifier
;
opt_member_list: opt_member
| opt_member_list opt_member
;
opt_member: decl_identifier ':' type ';'
;

```
type: std_type
    | class_head
    | set_type
    ;
set_type: set '[' decl_identifier ']'
    ;
set: SET
    ;
constraints: constraint
     | constraints constraint
    ;
constraint: CONSTRAINT sym_identifier const_decls const_constraint END
    ;
const_decls: const_decl
    | const_decls const_decl
    ;
const_decl: type decl_identifier ';'
    ;
const_constraint: const_expression
    | const_statements const_expression
    ;
const_expression: expression ';'
    | forEvery_expression '{'const_constraint '}'
    ;
const_statements:
    | const_statement
     | const_statements const_statement
    ;
const_statement: identifier assignop expression ';'
    | forEvery_expression '{' const_statements '}'
    ;
forEvery_expression:ForEvery expression
    ;
expression: simple_expression
    | simple_expression relop simple_expression
    ;
simple_expression: term
    | sign term
    | simple_expression addop term
    ;
term: factor
    | term mulop facto
    ;
factor: identifier
```

```
    | num
    | '(' expression ')'
    ;
sign: '+'
    | '-'
    ;
num: sym_integer
    | sym_float
    ;
relop: '>='
    | '>'
    | '<'
    | '<='
    | '=='
    | '!='
    | MUST_IN
    ;
assignop: '='
    | IN
    | NOT_IN
    ;
addop: '+'
    | '-'
    | OR
    ;
mulop: DIV
    | MOD
    | '*'
    | '/'
    | AND
    ;
std_type: STRING
    | FLOAT
    |INT
    ;
decl_identifier: sym_identifier
    ;
identifier: decl_identifier
    | identifier '.' decl_identifier
    ;
```

There are some examples for constraints in section 4.2.1.

2.1.2 the algorithm to generate demons

There are 2 steps to generate the demon code. First, split the classes along both paths in this constraint into two kinds of sets: AD (access-direct) and AP (access-path) sets. Here's an algorithm for this:

There are 2 paths in one predicate. And, for each path,

1. the leaves of the path belongs to AD set and the rest of it belongs to AP set.

2. if this node is a set type, its element type also belongs to the same set (AD or AP) with this set type.

For example, in a constraint "Client.reservations.number_of_persons <= Client. number_of_persons", the left path is "Client, SET[Reservation]" and the right path is "Client". Then, according to the above algorithm, the AD set for left path is {SET[Reservation], Reservation} since the data type of "Client->reservations" is SET [Reservation] and AP set is {Client}. For the right path, the AP is null and AD is {Client}. To give a more nested example, if we have the following classes:

```
class Courses {
                    _course-name: string;
                    _instructor: class Professor;
                    }
class Student {
                    _course-taken: set[Courses];
                    _student-name: string;
                    _student-number: int;
                    _age: int;
                    _name: string;
                    _parents: set[Person];
                    }
class Professor {
                    _age: int;
                    _name: string;
                    _courses-given: set[Courses];
                    }
class Person {
                    _age: int;
                    _name: string;
                    }
```

And now we declare "Student._parents._name!=Student._course-taken.instructor._name". In this example, the AD set for right path is {"Professor"} and AP set are {Student, SET[Courses], Courses}. And, the AD and AP set of left path is {SET[Person], Person} and {Student} accordingly.

After spliting the classes into AD and AP set, the procedures will be generated for each class according to which set it belongs to and if this class is a set type or not. The first part of the generated code is the header of this generated file (all the generated procedures are put in this file) and the constraint instance declaration. Section 2.1.3 will have a full description about the constraint instance declaration and the header file. The second part is the generated code which will be attached to those objects related to the constraint. The following is the algorithm telling what kind of codes should be generated:

1. if class A is a collection type and it's in AD: Since A is a set type and it's in AD, its parent node must be in AP and is not null. Assuming the parent of A is PAR and PAR.a is the object with type SET[A]. Here is the generated code that should be attached to PAR.a:

```
void PAR_a (A *element, int mod_status, CI_TYPE *ci) {
        if (mod_status == INSERT) {
                demonize (element, ci);
                ci->add_ckval? (element->ckattr());}
        if (mod_status == DELETE) {
                undemonize(element,ci);
                ci->del_ckval? (element->ckattr());}
        }
```

The "ci" is the constraint variable and the "ckattr" is the attribute that is supposed to be checked in the constraint. (For example, in the above example, ckattr is "_name") CI_TYPE is the data type for ci which can be decided when the constraint is imposed according to the algorithm in section 2.1.3. Besides, "?" is depends on which path "A" belongs to. If this is the left path, "?" is 1, otherwise, "?" is 2. For instance, the following is the code generated for SET[Person] which belongs to AD set of the left path:

```
void Student_parents (Person *element, int mod_status,C_4<string> *C1){
        if (mod_status == INSERT) {
                demonize(element,C1);
                C1->add_ckval1(element->name());
        }
        if (mod_status == DELETE) {
                undemonize(element,C1);
                C1->del_ckval1(element->name());
        }
}
```

2. if class A is in AD and not a collection type: the following is the generated code according to A:

```
void A_ckattr (Type ckattr, Type new_ckattr, CI_TYPE *ci) {
        if (new_ckattr op ci->ckval?)
                ci->new_ckval? (ckattr, new_ckattr);
        else
                exception();
}
```

The "op" is the operator in this constraint and the "?" is 1 if A is in the right path, otherwise it's 2. "Type" is the data type of the checked attribute. The rest is the same with the above. Here's the code generated from the left AD set "Person":

```
void Person_name (int _name, int new__name, C_4<string> *C1){
        if (new__name!=C1->ckval2)
                C1->new_ckval1(_name,new__name);
                else
        exception();
```

```
}
```

The following is the code generated from the right AD set "Professor":

```
void Professor_name (int _name, int new_name, C_4<string> *C1){
        if (new_name!=C1->ckval1)
                C1->new_ckval2(_name,new_name);
        else
                exception();
}
```

3. if class A is in AP and not a collection type: in this case, assuming child is the child node of A in the path and the corresponding attribute name is attribute (That is, A.attribute is an object with type child), the following code is generated:

```
void A_attribute (child *attribute, child *new_attribute) {
        SplicePath (attribute, new_attribute);
}
```

For example, the "Student" is in AP set of the left path and its child node is "Person" (the corresponding attribute name is _parents).Therefore, the following codes is generated:

```
void Student_parents (Person *_parents, Person *new_parents){
        SplicePath(_parents,new_parents);
}
```

Besides, "Student" is also in AP of the right path and its child node is "Courses" (the corresponding attribute name is _course-taken.):

```
void Student_course-taken (Courses *_course-taken, Courses *new_course-
taken){
        SplicePath (_course-taken, new_course-taken);
}
```

"Course" is also in AP and its child node is "Professor" (the corresponding attribute name is "_instructor"):

```
void Courses_instructor(Professor *_instructor, Professor *new_instruc-
tor){
        SplicePath(_instructor,new_instructor);
}
```

4. if class A is in AP and a collection type: assuming PAR is A's parent node and a is the corresponding attribute name of class A, the following code is generated:

```
void PAR_a (A *element, int mod_status, CI_TYPE *ci) {
        if (mod_status == INSERT) {
                demonize (element, ci);
        if (mod_status == DELETE) {
                undemonize (element, ci);
}
```

For example, "courses" is in AP of the right path and its parent node is "Student". The following code is generated:

```
void Student_course-taken (Courses *element, int mod_status, C_4<string>
*C1){
        if (mod_status == INSERT)
                demonize (element, C1);
        if (mod_status == DELETE)
                undemonize (element, C1);
}
```

2.1.3 constraint instances

In the algorithm above, we need a constraint instance to store all the checked value for both sides. Each side of the instance could be a collection or just single value.Therefore, we have 4 kinds of class definition for constraint instances:

```
template <class Type>
class C_1 {
    public:
        C_1();
        ~C_1();
        Type ckval1;
        Type ckval2;
        void new_ckval1(Type old, Type new);
        void new_ckval2(Type old, Type new);
}

template <class Type>
class C_2 {
   public:
        C_2();
        ~C_2();
        Type ckval1;
        os_Set <Type *> ckval2;
        void new_ckval1(Type old, Type new);
        void new_ckval2(Type old, Type new);
        void add_ckval2(Type a);
        void del_ckval2(Type a);
}

template <class type>
class C_3 {
    public:
        C_3();
        ~C_3();
        os_Set <Type *> ckval1;
        Type ckval1;
        void new_ckval1(Type old, Type new);
```

```
        void new_ckval2(Type old, Type new);
        void add_ckval1(Type a);
        void del_ckval1(Type a);
}

template <class Type>
class C_4 {
    public:
        C_4();
        ~C_4();
        os_Set <Type*> ckval1;
        os_Set <Type*> ckval2;
        void new_ckval1(Type old, Type new);
        void new_ckval2(Type old, Type new);
        void add_ckval1(Type a);
        void del_ckval1(Type a);
        void add_ckval2(Type a);
        void del_ckval2(Type a);
}
```

To impose a constraint, you must write the demonize function on your own to store all the checked values into the constraint instance. The system can not store the checked values automatically. The difficulties are discussed in section 2.3.1 and section 2.3.2 will talk about what should be written to impose a constraint, including installing the checked values into the constraint instance.

To initialize the constraint instance, we should also generate the code for the constraint instance declaration as we mentioned in the above section. Here's an algorithm to decide which kind of constraint instance should be declared:

1. if left AD is a set or left AP contains a set => left side of the instance should be a cache set
2. if left AP is empty and left AD is not a set and right AP is empty and right AD is not a set => both side of the instance should be cache sets
3. if right AD is a set or right AP contains a set => right side of the instance should be a cache set

Hence, the system will decide which kind of constraint should be initialized according to the above algorithm and declares the right type for the constraint instance in the functions generated in section 2.1.2.

To make the generated function code robust and able to be compiled, it's essential to add some headers to the generated file. This part is very trivial but needs to be modified from time to time:

```
#define TRUE 1
#define FALSE 0
#include <ostore/ostore.hh>
#include <ostore/coll.hh>
```

```
#include <iostream.h>
#include "schema.hh"
extern int operator == (int, os_Set <int*>);
extern int operator <= (int, os_Set <int*>);
extern int operator >= (int, os_Set <int*>);
extern int operator < (os_Set <int*>, int);
extern int operator > (int, os_Set <int*>);
extern int operator! = (int, os_Set <int*>);
extern void SplicePath (Reservation*, Reservation*);
extern void demonize (Reservation*, C_2<int>*);
extern void undemonize (Reservation*, C_2<int>*);
extern void exception();
```

For more details about how to modify the header, see section 2.3.2.

2.2. Database Implementation

The database we built is the database for a travel agency. It's built on top of ObjectStore. Section 2.2.1 gives a brief description about ObjectStore. Section 2.2.2 has a full description about how the data is organized.

2.2.1 work environment

Since this database is built on top of ObjectStore and utilize its built-in data types, it's essential to have a thorough understanding of ObjectStore. Therefore, in this section, we will see some basic concepts about ObjectStore:

1. ***How does ObjectStore organize its persistent data***:

The persistent data are stored in disks. A site can have several disks which contain one or more ObjectStore file systems. These file systems consist of ObjectStore databases. The ObjectStore databases consist of segments, which, in turn, consists of pages. The file systems in ObjectStore is similar to Unix file system. It organize the databases into directories. The hierarchy structure of directories is just like what Unix does to its directories, but these two systems are independent. The organization is logical rather than physical - that means, two databases stored in the same directory may be stored in different file systems, and, two databases in different directories may be stored in the same file system.

2. ***processes to execute ObjectStore:***

There are three auxiliary processes for executing ObjectStore: ObjectStore server, directory manager and cache manager. Each site has one or more directory managers and ObjectStore servers. Each node running one or more ObjectStore applications has a cache manager.

The ObjectStore server controls all kinds of access to an ObjectStore file system, like storing the persistent data or retrieving the persistent data. Different database can be handled by different servers. Therefore, when an application involves several databases, it also invokes several ObjectStore servers.

When an ObjectStore application starts, it will invokes a cache manager. There can only be one cache manager on a machine. That means, even if there are several ObjectStore application running on a machine, only a single cache manager is running on that machine. The responsibility for the cache manager is to manage the application's client cache, a local holding area for data which has been mapped or waiting to be mapped into virtual memory.

3. Memory management:

In ObjectStore, all pointers in database applications are regular virtual memory pointers. A pointer that points to a persistent object initially has its value a virtual memory address that is not mapped. After the pointer is dereferenced by the application, the operating system signals a violation which is handled by ObjectStore. Then, the violation handler will find the intended persistent objects and performs the data transfer and mapping. The data transfer between database memory and program memory is automatic and transparent to the users. Therefore, we can see that the ObjectStore can detect the references in a running program to persistent objects, then automatically transfers the segments which the referenced objects reside in to the application's cache. Then, the page containing the referenced data is mapped into the virtual memory.

This mapping architecture allows every data type in ObjectStore to be either persistent or transient, no matter whether it's a user-defined type or C++ built-in type. Functions can accept either persistent or non-persistent data. Therefore, existing applications designed for use with transient data can also be used in ObjectStore. Moreover, it provides a higher speed of pointer processing. Once an object has been mapped into virtual memory, all pointers pointing to it can be treated as regular virtual memory pointer and be processed at regular hardware speed.

In order to achieve the above goals, ObjectStore must know the classes of objects and the layout of these classes for each database. Therefore, it needs to store schema information for each database. The process of generating the schema information has two stages. The first stage occurs at compile time and the second stage occurs while linking.

The first stage creates a compilation schema database. The uses of each class in the source files are stored in compilation schema database. For example, if there's a class serving as an entry point for retrieval from persistent storage, there's a corresponding object representing this class stored in the compilation schema database. Besides, for each class reachable by navigation from other classes, there's also a corresponding object representing this class stored in the compilation schema database.

The second stage creates an application schema database. In the application schema database, all objects in the compilation schema database are brought together with objects representing classes in the schemes of any libraries in the application.

The process of generating the schema information can be automated through the Make macros. Here's a Makefile example:

```
include $(OS_ROOTDIR)/etc/ostore.lib.mk
OS_COMPILATION_SCHEMA_DB_PATH= /$(LOGNAME)/db_comp.schema
OS_APPLICATION_SCHEMA_DB_PATH= /$(LOGNAME)/db_app.schema
LDLIBS = -los -loscol
SOURCES = init.cc mem_function.cc
OBJECTS = init.o
EXECUTABLES = init
# set CC to your C++ compiler command
```

```
# unset TFLAGS (+OSTD is an OSCC flag to allow only standard C++)
TFLAGS=
CCC=OSCC

all: ${EXECUTABLES}

init: init.o mem_function.o schema_standin
$(OS_LINK) $(TFLAGS) -o init init.o mem_function.o $(LDLIBS)
init.o: init.cc
${CCC} $(CPPFLAGS) $(TFLAGS) -c init.cc
mem_function.o: mem_function.cc
${CCC} $(CPPFLAGS) $(TFLAGS) -c mem_function.cc
schema_standin: schema.cc
OSCC $(CPPFLAGS) -batch_schema $(OS_COMPILATION_SCHEMA_DB_
PATH) schema.cc
touch schema_standin

clean:
        osrm -f ${OS_COMPILATION_SCHEMA_DB_PATH}
        rm -f ${EXECUTABLES} ${OBJECTS} schema_standin

.depend:
        osmakedep .depend $(CPPFLAGS) -files $(SOURCES)

include .depend
```

From the above makefile, we can see that the user must first write a schema.cc file to store the class information. Then, through the makefile, the ObjectStore will create /yfl/db_comp.schema and /yfl/db_app.schema files in ObjectStore disk, which, as we mentioned above, are the corresponding compilation schema database and application schema database. The following is a sample schema.cc file for my schema schema.hh (see appendix):

```
static void dummy () {
        OS_MARK_SCHEMA_TYPE(os_Set<Country *>);
        OS_MARK_SCHEMA_TYPE(os_Set<City *>);
        OS_MARK_SCHEMA_TYPE(os_Set<Person*>);
        OS_MARK_SCHEMA_TYPE(os_Set<Client *>);
        OS_MARK_SCHEMA_TYPE(os_Set<Client_employee *>);
        OS_MARK_SCHEMA_TYPE(os_Set<Employee *>);
        OS_MARK_SCHEMA_TYPE(os_Set<Hotel *>);
        OS_MARK_SCHEMA_TYPE(os_Set<Hotel_reservation *>);
        OS_MARK_SCHEMA_TYPE(os_Set<Monument *>);
        OS_MARK_SCHEMA_TYPE(os_Set<Museum *>);
        OS_MARK_SCHEMA_TYPE(os_Set<Place_to_go *>);
```

```
    OS_MARK_SCHEMA_TYPE(os_Set<Reservation *>);
    OS_MARK_SCHEMA_TYPE(os_Set<Theater *>);
    OS_MARK_SCHEMA_TYPE(os_Set<Tour *>);
    OS_MARK_SCHEMA_TYPE(os_Set<Tour_reservation *>);

    OS_MARK_SCHEMA_TYPE(Address);

    OS_MARK_SCHEMA_TYPE(City);
    OS_MARK_SCHEMA_TYPE(Client);
    OS_MARK_SCHEMA_TYPE(Client_employee);
    OS_MARK_SCHEMA_TYPE(Country);

    OS_MARK_SCHEMA_TYPE(Date);
    OS_MARK_SCHEMA_TYPE(Day_price);

    OS_MARK_SCHEMA_TYPE(Employee);

    OS_MARK_SCHEMA_TYPE(Hotel);
    OS_MARK_SCHEMA_TYPE(Hotel_reservation);
    OS_MARK_SCHEMA_TYPE(Monument);
    OS_MARK_SCHEMA_TYPE(Money);
    OS_MARK_SCHEMA_TYPE(Museum);

    OS_MARK_SCHEMA_TYPE(Person);
    OS_MARK_SCHEMA_TYPE(Place_to_go);

    OS_MARK_SCHEMA_TYPE(Reservation);

    OS_MARK_SCHEMA_TYPE(Stage);
    OS_MARK_SCHEMA_TYPE(string);

    OS_MARK_SCHEMA_TYPE(Theater);
    OS_MARK_SCHEMA_TYPE(Tour);
    OS_MARK_SCHEMA_TYPE(Tour_reserva thin);
}
```

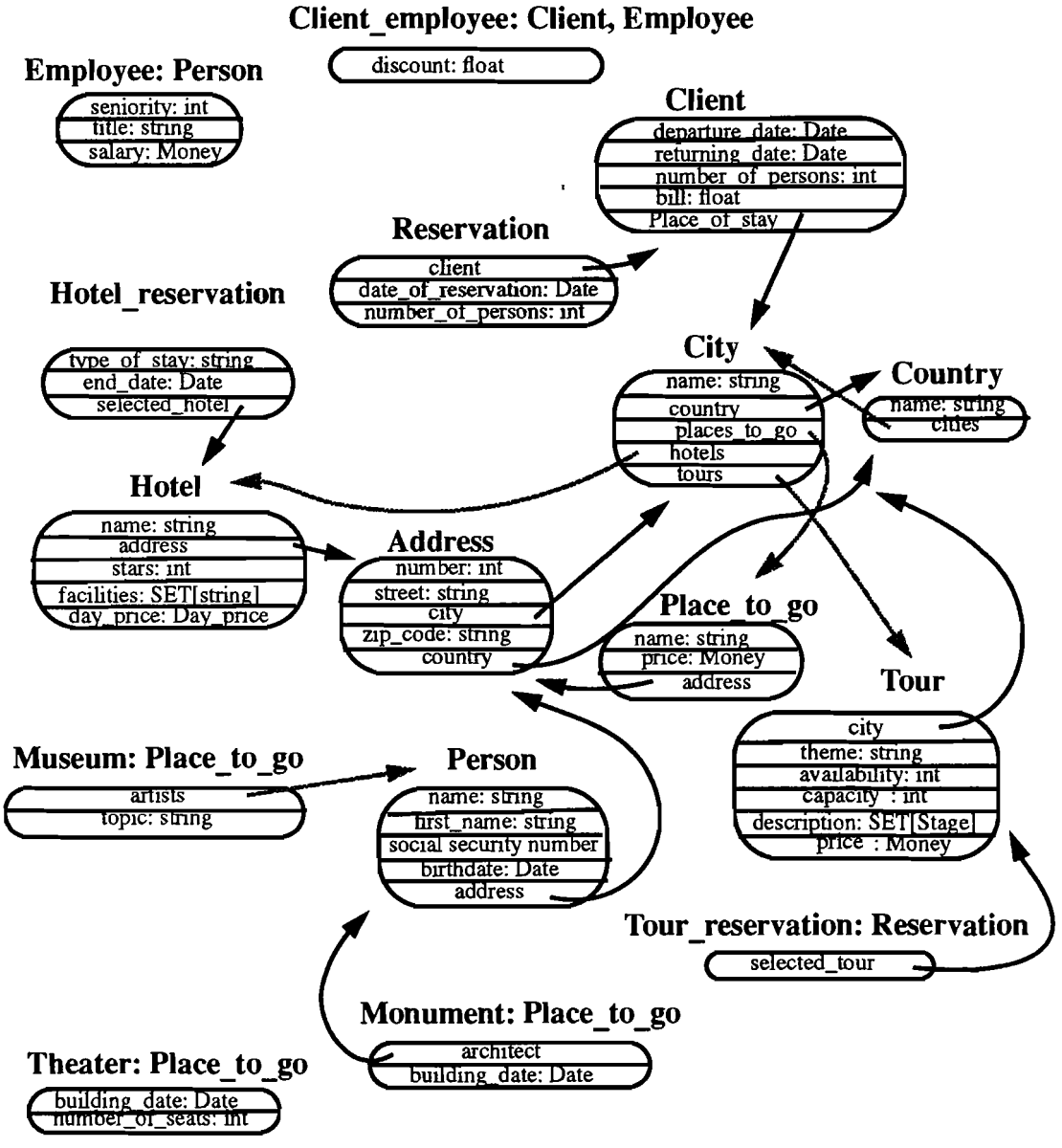
4. Programming Interfaces:

In ObjectStore, the user can use any of the following compilers to build his application: C compiler, C++ compiler with template class or without template class, and C++ compiler with template class, ObjectStore DDL and DML (this is called Object Design C++ compiler).Therefore, to use ObjectStore, you can use the C Library Interface, the C++ Library Interface without class templates, the C++ Library Interface with class tem-

plates or the C++ Library Interface with class templates and ObjectStore DML. All these approaches can be mixed freely. For example, a program can use both DML and C++ Library Interface to perform queries in the same function

2.2.2 data structure

We built a database for travel agency to impose a constraint on it. In this database, we use 15 persistent variables to be the entry points for those 15 kinds of class instances. For example, whenever we create a new Person instance, this instance will be added into the "os_Set <Person *> people" variable. Thereafter, we can find out any "Person instance" from the "people" bag. The following is our database schema: (the gray arrow line indicates the domain of this attribute value is a set type. "A: B" means A is a subtype of B.)


Client_employee: Client, Employee
discount: float
Employee: Person
seniority: int
title: string
salary: Money
Client
departure_date: Date
returning_date: Date
number_of_persons: int
bill: float
Place_of_stay
Reservation
client
date_of_reservation: Date
number_of_persons: int
Hotel_reservation
type_of_stay: string
end_date: Date
selected_hotel
City
name: string
country
places_to_go
hotels
tours
Country
name: string
cities
Hotel
name: string
address
stars: int
facilities: SET[string]
day_price: Day_price
Address
number: int
street: string
city
zip_code: string
country
Place_to_go
name: string
price: Money
address
Tour
city
theme: string
availability: int
capacity : int
description: SET[Stage]
price : Money
Museum: Place_to_go
artists
topic: string
Person
name: string
first_name: string
social security number
birthdate: Date
address
Tour_reservation: Reservation
selected_tour
Monument: Place_to_go
architect
building_date: Date
Theater: Place_to_go
building_date: Date
number_of_seats: int

2.3. Demonize & undemonize

This section describes the implementations about how to demonize and undemonize the generated code. Since there are some major difficulties in ObjectStore, the users have to write some additional functions on their own. In 2.3.1, we'll see what kind of difficulties it has. Then, section 2.3.2 describes the function specifications that needs to be modified. Finally, section 2.3.3 talks about how to impose/ disattach the generated codes into the objects.

2.3.1 difficulties in ObjectStore

There are two major difficulties in ObjectStore that result in the need of modifying the files and limit the type of constraint.

Though our constraint language is powerful enough to express any kind of constraint, the built-in operator of the collection type in ObjectStore limits the kinds of constraint that can be imposed on the database. For example, if we want to impose this constraint "No employee's salary is greater than his manager". Then our constraint instance must contains a collection type "os_Set <int>" and a single value type "int". Whenever we want to check if there's a constraint violation, sometimes we have to take out all the instances in the collection and compares their values with their manager's salary. Therefore, we expect that there's a set of operators for "os_Set<Type>" defined by ObjectStore that meets our needs. However, these built-in operators are insufficient for our constraint and sometimes cause some troubles. In our system, we do define a set of operators for (int, os_Set<int>) pairs. However, when it comes to (os_Set<int>, os_Set<int>) or (os_Set<int>, int), there might be some problems since ObjectStore have the built-in operators for them and these are definitely not the operation type we want. The other problem for our constraint type is, we can only check the integer or float values at this moment. If we want to check something like "if this object is in that set of objects", we not only have to define a new set of operators but also have to deal with the "object-equality" difficulty in object-oriented model. It might be able to be solved by dividing the problem into "deep-equal", "shallow-equal" and "equal-identity", etc. But it's too complicated and inefficient to define on your own.

The other difficulty in ObjectStore is we can not assign function pointers to our database since the real location of the procedures will change from time to time and it's nonsense to store the function pointer into the persistent memory. Therefore, we use an index to indicate if the constraint is imposed instead of function pointer. The drawback is, all the functions are predefined before we impose the constraint, therefore, when the constraint is imposed and generates the real function, this function is double defined. This takes a little bit of extra work to eliminate the old function.

Lastly, though the path for the constraint is known before demonization, we can not write a standard demonization/ undemonization function at this moment. Therefore, we have to modify the demonization/undemonization functions if a new constraint is imposed. The problem is that it is difficult to access persistent variables in ObjectStore automatically. In ObjectStore, you have to declare a persistent variable to be the entry point of some particular data. For example, if we have a set of persons in our database, we have to declare a persistent variable "persistent<db> os_Set <Person*> var_name". Thereafter, if the user want to get some particular person named "Stan", the variable "var_name" will act as an entry point to enable the programer to search this particular person named "Stan" by grabbing every person from set "var_name". This kind of character-

istic in ObjectStore prevents us from creating an automatic demonization process since we have no way to know what the variable's name is at compile time. However, there are some special cases that we can deal with. If the variable name is suffixed with "_set", we are able to build the demonization process automatically since in this case the variable name can be predicated at compiler time.

In order to build an automatic demonization process, the OODB should be able to access data without invoking any entry point or should provide standard persistent variable names as roots of the database (the entry points). Therefore, we can still attach the procedures without knowing the variable names at compile time and it will make it possible to build an automatic demonization process and make our system more robust and complete

2.3.2 modifying files

As mentioned in section 2.1.3, there are three files to be modified: /u/yfl/project/Dem_Attach/const_function.cc, /u/yfl/project/Dem_Attach/demonize.cc and /u/yfl/project/Dem_Attach/undemonize.cc.

/u/yfl/project/Dem_Attach/const_function.cc contains all the functions called by the main procedures generated by the constraint compiler, including the overloaded operator functions. The fundamental functions that need to be defined are "demonize (Type Var,...)", "undemonize (Type Var,...)", "exception()" and "SplicePath (Type Var,...)".

/u/yfl/project/Dem_Attach/demonize.cc is the main function that initializes the constraint instance, stores all the check values into it, sets the index value for those affected objects to "ATT_DEMON" (attach the codes to these objects) and sets the constraint pointers for those objects to the newly create constraint instance.

/u/yfl/project/Dem_Attach/undemonize.cc includes all the responsibilities regarding setting the constraint pointers to NULL and setting the index value of those objects to "NULL_DEMON" and delete the constraint instance.

Besides the files mentioned above, we have to modify the generated file also since the header varies with the kinds of constraint. For example, the constraint example in our Chapter 4 will create a new file called "C1.cc" which contains all the generated procedures and a set of "extern function declarations" that should be defined in "const_function.cc" file. These "extern function declarations" should be modified if there's another constraint is being imposed.

2.3.3 running the program

After modifying the above files and functions, type "make" to recompile the code for demonization/undemonization. The executable file are "demonize" and "undemonize".

Chapter 3: Time & Space Analysis

In this chapter, we analyze how much time it takes to do a transaction before and after the procedures are attached to the objects and how much additional spaces are used for the constraint instance to store the checked value. Besides that, we also list the execution time for attaching (disattaching) procedures to (from) objects.

command	real time	user time	system time
update after demonization	23:26.0	6:24.1	1:24.6
update before demonization (same transaction as above)	23:59.8	6:22.9	1:24.3
demonize	4.95	3.9	0.45
undemonize	4.65	3.9	0.4

Table 3.1

The first two columns in Table 3.1 presents the execution time for 10,000 transactions. The data in this table was collected using the Unix time command. After we impose the constraint "Client.reservations.number_of_persons <= Client.number_of_persons" to our travel agency's database (the schema is in Appendix A) and the system attached the procedures to the database objects, we do the transaction "update client's number_of_persons" 10,000 times. From the data in the first column of Table 3.1, we can see it takes 6:24.1 seconds to execute 10,000 transaction after the procedures are attached to the objects. After that, we take those procedures out of the objects and do the same transactions again. This time it takes only 6:22.9 seconds to do these 10,000 transactions (see the second column: update before demonization).From table 3.1, we can see that after the procedures are attached to the objects, to modify the objects takes 1~4 seconds more than without attaching the procedures on these objects. Besides that, from Table 3.1, we can also see how much time it costs to attach the procedures to the objects and take the procedures out of these objects.

To impose the constraint and do the demonization we have to allocate spaces to store the checked values, that is, we have to allocate some memory for the constraint instance. From the constraint declaration in section 2.1.3, we can see each constraint instance is 8 bytes since both "int" and os_Set<int>" take 4 bytes.

Therefore, we can see that attaching the generated procedures to the objects doesn't drastically affect the performance of the updates, even though it has to execute extra functions each time when it's being updated. Moreover, it takes only a little extra space to store the checked values.

Chapter 4: User's Guide

4.1. Preliminaries

4.1.1 hardware

Since this project is built on top of the ObjectStore and ObjectStore is installed in dulcimer, the user must log on dulcimer to get access to the database.

4.1.2 file system

There are three subdirectories in this project: /u/yfl/project/Dem_Gen, /u/yfl/project/osDB and /u/yfl/project/Dem_Attach. /u/yfl/project/Dem_Gen contains all the files regarding the constraint compiler and code generator. /u/yfl/project/osDB is the directory for the database system. All the commands and source codes for database creation, database modification are in this directory. /u/yfl/project/Dem_Attach contains all the files about the demonization and undemonization.

4.2. Code Generator

This section talks about how to generate the code and what file will be created according to the constraint name. In section 4.2.1, we will see some examples for the constraint expression. Section 4.2.2 describes how to input the constraint into the constraint compiler. Section 4.2.3 tell us what's the output file after parsing the constraint.

4.2.1 constraint language

The grammar for the constraint language is listed in section 2.1.1. Therefore, in this section, we'll look at some examples for the constraint expression.

Suppose we have a database for a banking system, we want to make a constraint saying, "For every Object with deposit type, its customer name must appear in some Object with customer type". Therefore, the following is the constraint expression for it:

example 1:

```
class customer {_customer-name: string;
                address: string;}
class deposit {_customer-name: string;}
constraint C2
class customer x;
class deposit y;
string _customer-name;
forEvery y {x._customer-name == y._customer-name;}
end
```

The constraint name for example 1 is "C2". And, assuming this constraint is stored as a file called "bank.con".

Suppose we have another database for a company and we want to declared "No employee's salary should be greater than his manager's salary", example 2 is the expression for this constraint:

example 2:

```
class Employee {_salary: int;}
class Manager {_salary: int;}
class Dept {_head: class Manager;
            _workers: set[Employee];}
constraint C3
class Dept x;
```

class Employee y;
int _salary;
forEvery y {y in x._workers;
y._salary<x._head._salary;}
end

The constraint name is "C3" and let's assume the file name is "com.con".

After writing down the constraint, we'll get to the part of sending the constraint file into the code generator.

4.2.2 input file

To send the constraint into the constraint compiler, type "const < filename".

4.2.3 output file

The generated file name is dependent on the constraint name we mentioned in section 4.2.1. For example, the output file name for example 1 and example 2 is "C2" and "C3" accordingly. We suggest using "*.cc" as the constraint name since the output file will be compiled as an ObjectStore file afterwards, as we mentioned in chapter 2 about demonization and undemonization.

4.3. Database

4.3.1 How to create a database

To create a database for traveling agency, type "/u/yfl/project/init filename". Make sure the filename including the whole path like /yfl/project/db or /yfl/db, etc. Notice the system in ObjectStore is very similar but a little different from Unix system.

4.3.2 How to get access to an existed database

To get access to an existed database, type "dbms filename".

4.4. Example

The following is dumped from testing script file:

```
Script started on Mon Jun 22 14:50:46 1992
(dulcimer)1--> ../Dem_Attach/undemonize /yfl/project/db


(dulcimer)2--> dbms /yfl/project/db


**************************
* 1 | create new object *
*------------------------*
* 2 | modify data *
*------------------------*
* 3 | retrieve data *
*------------------------*
* 4 | delete objects *
*------------------------*
```

```
* 0 | exit *
****************************
 enter your selection : 2
*****************************************************
* 1 | modify a person instance *
*-------------------------------------------*
* 2 | modify a city instance *
*-------------------------------------------*
* 3 | modify a country instance *
*-------------------------------------------*
* 4 | modify a client instance *
*-------------------------------------------*
* 5 | modify a client_employee instance *
*-------------------------------------------*
* 6 | modify an employee instance *
*-------------------------------------------*
* 7 | modify a hotel instance *
*-------------------------------------------*
* 8 | modify a reservation for hotel *
*-------------------------------------------*
* 9 | modify a monument instance *
*-------------------------------------------*
* 10 | modify a museum instance *
*-------------------------------------------*
* 11 | modify a reservation instance *
*-------------------------------------------*
* 12 | modify a place to go instance *
*-------------------------------------------*
* 13 | modify a theater instance *
*-------------------------------------------*
* 14 | modify a tour *
*-------------------------------------------*
* 15 | modify a tour reservation *
*****************************************************

 enter your selection :4
 enter the social_security number of the client :035-60-7738
***********************************
* 1 | social security number *
*---------------------------------*
* 2 | last name *
*---------------------------------*
* 3 | first name *
*---------------------------------*
```

```
* 4 | birthdate *
*-------------------------------*
* 5 | address *
*-------------------------------*
* 6 | reservations *
*-------------------------------*
* 7 | place of stay *
*-------------------------------*
* 8 | departure date *
*-------------------------------*
* 9 | returning date *
*-------------------------------*
* 10| number of persons *
*-------------------------------*
* 11| bill *
************************************
 selection : 10
 enter the new number of persons : 2
****************************
* 1 | create new object *
*-------------------------*
* 2 | modify data *
*-------------------------*
* 3 | retrieve data *
*-------------------------*
* 4 | delete objects *
*-------------------------*
* 0 | exit *
****************************
 enter your selection : 3
******************************************************
* 1 | get all the person instance *
*-----------------------------------------------*
* 2 | get all the city instance *
*-----------------------------------------------*
* 3 | get all the country instance *
*-----------------------------------------------*
* 4 | get all the client instance *
*-----------------------------------------------*
* 5 | get all the client_employee instance *
*-----------------------------------------------*
* 6 | get all the employee instance *
*-----------------------------------------------*
* 7 | get all the hotel instance *
```

```
*-----------------------------------------------*
* 8 | get all the reservation for hotel *
*-----------------------------------------------*
* 9 | get all the monument instance *
*-----------------------------------------------*
* 10 | get all the museum instance *
*-----------------------------------------------*
* 11 | get all the reservation instance *
*-----------------------------------------------*
* 12 | get all the place to go instance *
*-----------------------------------------------*
* 13 | get all the theater instance *
*-----------------------------------------------*
* 14 | get all the tour *
*-----------------------------------------------*
* 15 | get all the tour reservation *
*************************************************

 enter your selection :4
NAME          SC#    BIRTHDATEADDRESS
*************************************************************************
**************
Li , Yu-Fang035-60-7738April 11 , 19679 Alumni Ave.
                      Providence,02906
                      USA

SC#    # OF PERSONSPLACE TO STAYDEP DATERETURN DATE BILL
*************************************************************************
**************
035-60-77382 New YorkMay 18 , 1992 MondayMay 25 , 1992 Monday500.000000
 display the reservations for some client ?(yes or no)yes
 enter the social security # of the client :035-60-7738
 reservation information for this person :
SC# of CLIENT RESERVATION DATE NUMBER OF PERSON
*************************************************************************
****
035-60-7738May 19 , 19922
035-60-7738May 20 , 19923
035-60-7738May 21 , 19924
035-60-7738May 22 , 19925
 enter the other client ?(yes or no)no
*****************************
* 1 | create new object *
*-------------------------*
```

```
* 2 | modify data *
*-------------------------*
* 3 | retrieve data *
*-------------------------*
* 4 | delete objects *
*-------------------------*
* 0 | exit *
*****************************
 enter your selection : 2
*********************************************************
* 1 | modify a person instance *
*-----------------------------------------------*
* 2 | modify a city instance *
*-----------------------------------------------*
* 3 | modify a country instance *
*-----------------------------------------------*
* 4 | modify a client instance *
*-----------------------------------------------*
* 5 | modify a client_employee instance *
*-----------------------------------------------*
* 6 | modify an employee instance *
*-----------------------------------------------*
* 7 | modify a hotel instance *
*-----------------------------------------------*
* 8 | modify a reservation for hotel *
*-----------------------------------------------*
* 9 | modify a monument instance *
*-----------------------------------------------*
* 10 | modify a museum instance *
*-----------------------------------------------*
* 11 | modify a reservation instance *
*-----------------------------------------------*
* 12 | modify a place to go instance *
*-----------------------------------------------*
* 13 | modify a theater instance *
*-----------------------------------------------*
* 14 | modify a tour *
*-----------------------------------------------*
* 15 | modify a tour reservation *
*********************************************************

 enter your selection :4
 enter the social_security number of the client :035-60-7738
************************************
```

```
* 1 | social security number *
*--------------------------------*
* 2 | last name *
*--------------------------------*
* 3 | first name *
*--------------------------------*
* 4 | birthdate *
*--------------------------------*
* 5 | address *
*--------------------------------*
* 6 | reservations *
*--------------------------------*
* 7 | place of stay *
*--------------------------------*
* 8 | departure date *
*--------------------------------*
* 9 | returning date *
*--------------------------------*
* 10| number of persons *
*--------------------------------*
* 11| bill *
************************************
 selection : 10
 enter the new number of persons : 20
****************************
* 1 | create new object *
*-------------------------*
* 2 | modify data *
*-------------------------*
* 3 | retrieve data *
*-------------------------*
* 4 | delete objects *
*-------------------------*
* 0 | exit *
****************************
 enter your selection : 3
******************************************************
* 1 | get all the person instance *
*---------------------------------------------*
* 2 | get all the city instance *
*---------------------------------------------*
* 3 | get all the country instance *
*---------------------------------------------*
* 4 | get all the client instance *
```

```
*-----------------------------------------------*
* 5 | get all the client_employee instance *
*-----------------------------------------------*
* 6 | get all the employee instance *
*-----------------------------------------------*
* 7 | get all the hotel instance *
*-----------------------------------------------*
* 8 | get all the reservation for hotel *
*-----------------------------------------------*
* 9 | get all the monument instance *
*-----------------------------------------------*
* 10 | get all the museum instance *
*-----------------------------------------------*
* 11 | get all the reservation instance *
*-----------------------------------------------*
* 12 | get all the place to go instance *
*-----------------------------------------------*
* 13 | get all the theater instance *
*-----------------------------------------------*
* 14 | get all the tour *
*-----------------------------------------------*
* 15 | get all the tour reservation *
************************************************

 enter your selection :4
NAME          SC#    BIRTHDATEADDRESS
********************************************************************************
**************
Li , Yu-Fang035-60-7738April 11 , 19679 Alumni Ave.
                      Providence,02906
                      USA

SC#    # OF PERSONSPLACE TO STAYDEP DATERETURN DATE BILL
********************************************************************************
**************
035-60-773820New YorkMay 18 , 1992 MondayMay 25 , 1992 Monday500.000000
 display the reservations for some client ?(yes or no)yes
 enter the social security # of the client :035-60-7738
 reservation information for this person :
SC# of CLIENT RESERVATION DATE NUMBER OF PERSON
********************************************************************************
****
035-60-7738May 19 , 19922
035-60-7738May 20 , 19923
```

```
035-60-7738May 21 , 19924
035-60-7738May 22 , 19925
 enter the other client ?(yes or no)no
*****************************
* 1 | create new object *
*-------------------------*
* 2 | modify data *
*-------------------------*
* 3 | retrieve data *
*-------------------------*
* 4 | delete objects *
*-------------------------*
* 0 | exit *
*****************************
 enter your selection : 0
(dulcimer)3--> ../Dem_Attach/demonize /yfl/project/db


ckval1=20
ckval2=2
ckval2=3
ckval2=4
ckval2=5
(dulcimer)4--> dbms /yfl/project/db


*****************************
* 1 | create new object *
*-------------------------*
* 2 | modify data *
*-------------------------*
* 3 | retrieve data *
*-------------------------*
* 4 | delete objects *
*-------------------------*
* 0 | exit *
*****************************
 enter your selection : 2
*****************************************************
* 1 | modify a person instance *
*-----------------------------------------------*
* 2 | modify a city instance *
*-----------------------------------------------*
* 3 | modify a country instance *
```

```
*-----------------------------------------------*
* 4 | modify a client instance *
*-----------------------------------------------*
* 5 | modify a client_employee instance *
*-----------------------------------------------*
* 6 | modify an employee instance *
*-----------------------------------------------*
* 7 | modify a hotel instance *
*-----------------------------------------------*
* 8 | modify a reservation for hotel *
*-----------------------------------------------*
* 9 | modify a monument instance *
*-----------------------------------------------*
* 10 | modify a museum instance *
*-----------------------------------------------*
* 11 | modify a reservation instance *
*-----------------------------------------------*
* 12 | modify a place to go instance *
*-----------------------------------------------*
* 13 | modify a theater instance *
*-----------------------------------------------*
* 14 | modify a tour *
*-----------------------------------------------*
* 15 | modify a tour reservation *
*************************************************

 enter your selection :4
 enter the social_security number of the client :035-60-7738
***********************************
* 1 | social security number *
*---------------------------------*
* 2 | last name *
*---------------------------------*
* 3 | first name *
*---------------------------------*
* 4 | birthdate *
*---------------------------------*
* 5 | address *
*---------------------------------*
* 6 | reservations *
*---------------------------------*
* 7 | place of stay *
*---------------------------------*
* 8 | departure date *
```

```
*---------------------------------*
* 9 | returning date *
*---------------------------------*
* 10| number of persons *
*---------------------------------*
* 11| bill *
************************************
 selection : 10
 enter the new number of persons : 2
constraint violation!
(dulcimer)5--> dbms /yfl/project/db


****************************
* 1 | create new object *
*-------------------------*
* 2 | modify data *
*-------------------------*
* 3 | retrieve data *
*-------------------------*
* 4 | delete objects *
*-------------------------*
* 0 | exit *
****************************
 enter your selection : 3
*****************************************************
* 1 | get all the person instance *
*-----------------------------------------------*
* 2 | get all the city instance *
*-----------------------------------------------*
* 3 | get all the country instance *
*-----------------------------------------------*
* 4 | get all the client instance *
*-----------------------------------------------*
* 5 | get all the client_employee instance *
*-----------------------------------------------*
* 6 | get all the employee instance *
*-----------------------------------------------*
* 7 | get all the hotel instance *
*-----------------------------------------------*
* 8 | get all the reservation for hotel *
*-----------------------------------------------*
* 9 | get all the monument instance *
*-----------------------------------------------*
```

```
* 10 | get all the museum instance *
*-----------------------------------------------*
* 11 | get all the reservation instance *
*-----------------------------------------------*
* 12 | get all the place to go instance *
*-----------------------------------------------*
* 13 | get all the theater instance *
*-----------------------------------------------*
* 14 | get all the tour *
*-----------------------------------------------*
* 15 | get all the tour reservation *
*************************************************

 enter your selection :4
NAME          SC#    BIRTHDATEADDRESS
*************************************************************************
**************
Li , Yu-Fang035-60-7738April 11 , 19679 Alumni Ave.
                      Providence,02906
                      USA

SC#    # OF PERSONSPLACE TO STAYDEP DATERETURN DATE BILL
*************************************************************************
**************
035-60-773820New YorkMay 18 , 1992 MondayMay 25 , 1992 Monday500.000000
 display the reservations for some client ?(yes or no)yes
 enter the social security # of the client :035-60-7738
 reservation information for this person :
SC# of CLIENT RESERVATION DATE NUMBER OF PERSON
*************************************************************************
****
035-60-7738May 19 , 19922
035-60-7738May 20 , 19923
035-60-7738May 21 , 19924
035-60-7738May 22 , 19925
 enter the other client ?(yes or no)no
*****************************
* 1 | create new object *
*-------------------------*
* 2 | modify data *
*-------------------------*
* 3 | retrieve data *
*-------------------------*
* 4 | delete objects *
```

```
*-------------------------*
* 0 | exit *
*****************************
 enter your selection : 2
*****************************************************
* 1 | modify a person instance *
*-----------------------------------------------*
* 2 | modify a city instance *
*-----------------------------------------------*
* 3 | modify a country instance *
*-----------------------------------------------*
* 4 | modify a client instance *
*-----------------------------------------------*
* 5 | modify a client_employee instance *
*-----------------------------------------------*
* 6 | modify an employee instance *
*-----------------------------------------------*
* 7 | modify a hotel instance *
*-----------------------------------------------*
* 8 | modify a reservation for hotel *
*-----------------------------------------------*
* 9 | modify a monument instance *
*-----------------------------------------------*
* 10 | modify a museum instance *
*-----------------------------------------------*
* 11 | modify a reservation instance *
*-----------------------------------------------*
* 12 | modify a place to go instance *
*-----------------------------------------------*
* 13 | modify a theater instance *
*-----------------------------------------------*
* 14 | modify a tour *
*-----------------------------------------------*
* 15 | modify a tour reservation *
*****************************************************

 enter your selection:11
 enter the client's social security number : 035-60-7738
 enter the date of reservation:
 enter the week :Tuesday
 enter the month :May
 enter the date :19
 enter the year :1992
*********************************
```

```
* 1 | client *
*-------------------------------*
* 2 | date of reservation *
*-------------------------------*
* 3 | number of persons *
**********************************
3
 enter the new number of persons : 40
constraint violation!
(dulcimer)6--> dbms /yfl/project/db


****************************
* 1 | create new object *
*-------------------------*
* 2 | modify data *
*-------------------------*
* 3 | retrieve data *
*-------------------------*
* 4 | delete objects *
*-------------------------*
* 0 | exit *
****************************
 enter your selection : 3
*****************************************************
* 1 | get all the person instance *
*-----------------------------------------------*
* 2 | get all the city instance *
*-----------------------------------------------*
* 3 | get all the country instance *
*-----------------------------------------------*
* 4 | get all the client instance *
*-----------------------------------------------*
* 5 | get all the client_employee instance *
*-----------------------------------------------*
* 6 | get all the employee instance *
*-----------------------------------------------*
* 7 | get all the hotel instance *
*-----------------------------------------------*
* 8 | get all the reservation for hotel *
*-----------------------------------------------*
* 9 | get all the monument instance *
*-----------------------------------------------*
* 10 | get all the museum instance *
```

```
*-----------------------------------------------*
* 11 | get all the reservation instance *
*-----------------------------------------------*
* 12 | get all the place to go instance *
*-----------------------------------------------*
* 13 | get all the theater instance *
*-----------------------------------------------*
* 14 | get all the tour *
*-----------------------------------------------*
* 15 | get all the tour reservation *
*************************************************

 enter your selection :4
NAME          SC#    BIRTHDATEADDRESS
*************************************************************************
**************
Li , Yu-Fang035-60-7738April 11 , 19679 Alumni Ave.
                      Providence,02906
                      USA

SC#    # OF PERSONSPLACE TO STAYDEP DATERETURN DATE BILL
*************************************************************************
**************
035-60-773820New YorkMay 18 , 1992 MondayMay 25 , 1992 Monday500.000000
 display the reservations for some client ?(yes or no)yes
 enter the social security # of the client :035-60-7738
 reservation information for this person :
SC# of CLIENT RESERVATION DATE NUMBER OF PERSON
*************************************************************************
****
035-60-7738May 19 , 19922
035-60-7738May 20 , 19923
035-60-7738May 21 , 19924
035-60-7738May 22 , 19925
 enter the other client ?(yes or no)no
*****************************
* 1 | create new object *
*-------------------------*
* 2 | modify data *
*-------------------------*
* 3 | retrieve data *
*-------------------------*
* 4 | delete objects *
*-------------------------*
```

```
* 0 | exit *
**************************
 enter your selection : 3
*****************************************************
* 1 | get all the person instance *
*-----------------------------------------------*
* 2 | get all the city instance *
*-----------------------------------------------*
* 3 | get all the country instance *
*-----------------------------------------------*
* 4 | get all the client instance *
*-----------------------------------------------*
* 5 | get all the client_employee instance *
*-----------------------------------------------*
* 6 | get all the employee instance *
*-----------------------------------------------*
* 7 | get all the hotel instance *
*-----------------------------------------------*
* 8 | get all the reservation for hotel *
*-----------------------------------------------*
* 9 | get all the monument instance *
*-----------------------------------------------*
* 10 | get all the museum instance *
*-----------------------------------------------*
* 11 | get all the reservation instance *
*-----------------------------------------------*
* 12 | get all the place to go instance *
*-----------------------------------------------*
* 13 | get all the theater instance *
*-----------------------------------------------*
* 14 | get all the tour *
*-----------------------------------------------*
* 15 | get all the tour reservation *
*************************************************

 enter your selection :4
NAME          SC#    BIRTHDATEADDRESS
*************************************************************************
**************
Li , Yu-Fang035-60-7738April 11 , 19679 Alumni Ave.
                     Providence,02906
                     USA

SC#    # OF PERSONSPLACE TO STAYDEP DATERETURN DATE BILL
```

```
**********************************************************************
**************
035-60-773820New YorkMay 18 , 1992 MondayMay 25 , 1992 Monday500.000000
 display the reservations for some client ?(yes or no)yes
 enter the social security # of the client :035-60-7738
 reservation information for this person :
SC# of CLIENT RESERVATION DATE NUMBER OF PERSON
**********************************************************************
****
035-60-7738May 19 , 19922
035-60-7738May 20 , 19923
035-60-7738May 21 , 19924
035-60-7738May 22 , 19925
 enter the other client ?(yes or no)no
*****************************
* 1 | create new object *
*-------------------------*
* 2 | modify data *
*-------------------------*
* 3 | retrieve data *
*-------------------------*
* 4 | delete objects *
*-------------------------*
* 0 | exit *
*****************************
 enter your selection : 2
*****************************************************
* 1 | modify a person instance *
*-----------------------------------------------*
* 2 | modify a city instance *
*-----------------------------------------------*
* 3 | modify a country instance *
*-----------------------------------------------*
* 4 | modify a client instance *
*-----------------------------------------------*
* 5 | modify a client_employee instance *
*-----------------------------------------------*
* 6 | modify an employee instance *
*-----------------------------------------------*
* 7 | modify a hotel instance *
*-----------------------------------------------*
* 8 | modify a reservation for hotel *
*-----------------------------------------------*
* 9 | modify a monument instance *
```

```
*-----------------------------------------------*
* 10 | modify a museum instance *
*-----------------------------------------------*
* 11 | modify a reservation instance *
*-----------------------------------------------*
* 12 | modify a place to go instance *
*-----------------------------------------------*
* 13 | modify a theater instance *
*-----------------------------------------------*
* 14 | modify a tour *
*-----------------------------------------------*
* 15 | modify a tour reservation *
*************************************************

 enter your selection :4
 enter the social_security number of the client :035-60-7738
**********************************
* 1 | social security number *
*--------------------------------*
* 2 | last name *
*--------------------------------*
* 3 | first name *
*--------------------------------*
* 4 | birthdate *
*--------------------------------*
* 5 | address *
*--------------------------------*
* 6 | reservations *
*--------------------------------*
* 7 | place of stay *
*--------------------------------*
* 8 | departure date *
*--------------------------------*
* 9 | returning date *
*--------------------------------*
* 10| number of persons *
*--------------------------------*
* 11| bill *
**********************************
 selection : 6
************************
* 1 | insert *
*----------------------*
* 2 | delete *
```

```
************************
 selection : 1
 enter the reservation you want to delete(or insert) :
 enter the client :
 enter the social_security number :
035-60-7738
 the client already exsist in the database
 enter the reservation date :
 enter the week :Monday
 enter the month :June
 enter the date :22
 enter the year :1992
 enter the number of persons :
10
*****************************
* 1 | create new object *
*-------------------------*
* 2 | modify data *
*-------------------------*
* 3 | retrieve data *
*-------------------------*
* 4 | delete objects *
*-------------------------*
* 0 | exit *
*****************************
 enter your selection : 3
*********************************************************
* 1 | get all the person instance *
*-----------------------------------------------*
* 2 | get all the city instance *
*-----------------------------------------------*
* 3 | get all the country instance *
*-----------------------------------------------*
* 4 | get all the client instance *
*-----------------------------------------------*
* 5 | get all the client_employee instance *
*-----------------------------------------------*
* 6 | get all the employee instance *
*-----------------------------------------------*
* 7 | get all the hotel instance *
*-----------------------------------------------*
* 8 | get all the reservation for hotel *
*-----------------------------------------------*
* 9 | get all the monument instance *
```

```
*-----------------------------------------------*
* 10 | get all the museum instance *
*-----------------------------------------------*
* 11 | get all the reservation instance *
*-----------------------------------------------*
* 12 | get all the place to go instance *
*-----------------------------------------------*
* 13 | get all the theater instance *
*-----------------------------------------------*
* 14 | get all the tour *
*-----------------------------------------------*
* 15 | get all the tour reservation *
************************************************

 enter your selection :4
NAME          SC#    BIRTHDATEADDRESS
********************************************************************************
**************
Li , Yu-Fang035-60-7738April 11 , 19679 Alumni Ave.
                      Providence,02906
                      USA

SC#    # OF PERSONSPLACE TO STAYDEP DATERETURN DATE BILL
********************************************************************************
**************
035-60-773820New YorkMay 18 , 1992 MondayMay 25 , 1992 Monday500.000000
 display the reservations for some client ?(yes or no)yes
 enter the social security # of the client :035-60-7738
 reservation information for this person :
SC# of CLIENT RESERVATION DATE NUMBER OF PERSON
********************************************************************************
****
035-60-7738May 22 , 19925
035-60-7738May 20 , 19923
035-60-7738June 22 , 199210
035-60-7738May 19 , 19922
035-60-7738May 21 , 19924
 enter the other client ?(yes or no)no
*****************************
* 1 | create new object *
*-------------------------*
* 2 | modify data *
*-------------------------*
* 3 | retrieve data *
```

```
*-------------------------*
* 4 | delete objects *
*-------------------------*
* 0 | exit *
*****************************
 enter your selection : 2
*****************************************************
* 1 | modify a person instance *
*-----------------------------------------------*
* 2 | modify a city instance *
*-----------------------------------------------*
* 3 | modify a country instance *
*-----------------------------------------------*
* 4 | modify a client instance *
*-----------------------------------------------*
* 5 | modify a client_employee instance *
*-----------------------------------------------*
* 6 | modify an employee instance *
*-----------------------------------------------*
* 7 | modify a hotel instance *
*-----------------------------------------------*
* 8 | modify a reservation for hotel *
*-----------------------------------------------*
* 9 | modify a monument instance *
*-----------------------------------------------*
* 10 | modify a museum instance *
*-----------------------------------------------*
* 11 | modify a reservation instance *
*-----------------------------------------------*
* 12 | modify a place to go instance *
*-----------------------------------------------*
* 13 | modify a theater instance *
*-----------------------------------------------*
* 14 | modify a tour *
*-----------------------------------------------*
* 15 | modify a tour reservation *
*****************************************************

 enter your selection:4
 enter the social_security number of the client :035-60-7738
************************************
* 1 | social security number *
*--------------------------------*
* 2 | last name *
```

```
*---------------------------------*
* 3 | first name *
*---------------------------------*
* 4 | birthdate *
*---------------------------------*
* 5 | address *
*---------------------------------*
* 6 | reservations *
*---------------------------------*
* 7 | place of stay *
*---------------------------------*
* 8 | departure date *
*---------------------------------*
* 9 | returning date *
*---------------------------------*
* 10| number of persons *
*---------------------------------*
* 11| bill *
************************************
 selection : 10
 enter the new number of persons : 9
constraint violation!
(dulcimer)7--> dbms /yfl/project/db


****************************
* 1 | create new object *
*-------------------------*
* 2 | modify data *
*-------------------------*
* 3 | retrieve data *
*-------------------------*
* 4 | delete objects *
*-------------------------*
* 0 | exit *
****************************
 enter your selection : 3
************************************************
* 1 | get all the person instance *
*---------------------------------------------*
* 2 | get all the city instance *
*---------------------------------------------*
* 3 | get all the country instance *
*---------------------------------------------*
```

```
* 4 | get all the client instance *
*-----------------------------------------------*
* 5 | get all the client_employee instance *
*-----------------------------------------------*
* 6 | get all the employee instance *
*-----------------------------------------------*
* 7 | get all the hotel instance *
*-----------------------------------------------*
* 8 | get all the reservation for hotel *
*-----------------------------------------------*
* 9 | get all the monument instance *
*-----------------------------------------------*
* 10 | get all the museum instance *
*-----------------------------------------------*
* 11 | get all the reservation instance *
*-----------------------------------------------*
* 12 | get all the place to go instance *
*-----------------------------------------------*
* 13 | get all the theater instance *
*-----------------------------------------------*
* 14 | get all the tour *
*-----------------------------------------------*
* 15 | get all the tour reservation *
*************************************************

 enter your selection :4
NAME          SC#    BIRTHDATEADDRESS
*************************************************************************
**************
Li , Yu-Fang035-60-7738April 11 , 19679 Alumni Ave.
                      Providence,02906
                      USA

SC#    # OF PERSONSPLACE TO STAYDEP DATERETURN DATE BILL
*************************************************************************
**************
035-60-773820New YorkMay 18 , 1992 MondayMay 25 , 1992 Monday500.000000
 display the reservations for some client ?(yes or no)yes
 enter the social security # of the client :035-60-7738
 reservation information for this person :
SC# of CLIENT RESERVATION DATE NUMBER OF PERSON
*************************************************************************
****
035-60-7738May 22 , 19925
```

```
035-60-7738May 20 , 19923
035-60-7738June 22 , 199210
035-60-7738May 19 , 19922
035-60-7738May 21 , 19924
 enter the other client ?(yes or no)no
**************************
* 1 | create new object *
*-------------------------*
* 2 | modify data *
*-------------------------*
* 3 | retrieve data *
*-------------------------*
* 4 | delete objects *
*-------------------------*
* 0 | exit *
**************************
 enter your selection : 2
*****************************************************
* 1 | modify a person instance *
*-----------------------------------------------*
* 2 | modify a city instance *
*-----------------------------------------------*
* 3 | modify a country instance *
*-----------------------------------------------*
* 4 | modify a client instance *
*-----------------------------------------------*
* 5 | modify a client_employee instance *
*-----------------------------------------------*
* 6 | modify an employee instance *
*-----------------------------------------------*
* 7 | modify a hotel instance *
*-----------------------------------------------*
* 8 | modify a reservation for hotel *
*-----------------------------------------------*
* 9 | modify a monument instance *
*-----------------------------------------------*
* 10 | modify a museum instance *
*-----------------------------------------------*
* 11 | modify a reservation instance *
*-----------------------------------------------*
* 12 | modify a place to go instance *
*-----------------------------------------------*
* 13 | modify a theater instance *
*-----------------------------------------------*
```

```
* 14 | modify a tour *
*-----------------------------------------------*
* 15 | modify a tour reservation *
*************************************************

 enter your selection :11
 enter the client's social security number : 035-60-7738
 enter the date of reservation:
 enter the week :Monday
 enter the month :June
 enter the date :22
 enter the year :1992
**********************************
* 1 | client *
*--------------------------------*
* 2 | date of reservation *
*--------------------------------*
* 3 | number of persons *
**********************************
3
 enter the new number of persons : 40
constraint violation!
(dulcimer)8--> dbms /yfl/project/db


****************************
* 1 | create new object *
*--------------------------*
* 2 | modify data *
*--------------------------*
* 3 | retrieve data *
*--------------------------*
* 4 | delete objects *
*--------------------------*
* 0 | exit *
****************************
 enter your selection : 3
*****************************************************
* 1 | get all the person instance *
*-----------------------------------------------*
* 2 | get all the city instance *
*-----------------------------------------------*
* 3 | get all the country instance *
*-----------------------------------------------*
```

```
* 4 | get all the client instance *
*-----------------------------------------------*
* 5 | get all the client_employee instance *
*-----------------------------------------------*
* 6 | get all the employee instance *
*-----------------------------------------------*
* 7 | get all the hotel instance *
*-----------------------------------------------*
* 8 | get all the reservation for hotel *
*-----------------------------------------------*
* 9 | get all the monument instance *
*-----------------------------------------------*
* 10 | get all the museum instance *
*-----------------------------------------------*
* 11 | get all the reservation instance *
*-----------------------------------------------*
* 12 | get all the place to go instance *
*-----------------------------------------------*
* 13 | get all the theater instance *
*-----------------------------------------------*
* 14 | get all the tour *
*-----------------------------------------------*
* 15 | get all the tour reservation *
*************************************************

 enter your selection :4
NAME          SC#    BIRTHDATEADDRESS
*********************************************************************************
**************
Li , Yu-Fang035-60-7738April 11 , 19679 Alumni Ave.
                      Providence,02906
                      USA

SC#    # OF PERSONSPLACE TO STAYDEP DATERETURN DATE BILL
*********************************************************************************
**************
035-60-773820New YorkMay 18 , 1992 MondayMay 25 , 1992 Monday500.000000
 display the reservations for some client ?(yes or no)yes
 enter the social security # of the client :035-60-7738
 reservation information for this person :
SC# of CLIENT RESERVATION DATE NUMBER OF PERSON
*********************************************************************************
****
035-60-7738May 22 , 19925
```

035-60-7738May 20 , 19923
035-60-7738June 22 , 199210
035-60-7738May 19 , 19922
035-60-7738May 21 , 19924
enter the other client ?(yes or no)no
****
* 1 | create new object *
----
* 2 | modify data *
----
* 3 | retrieve data *
----
* 4 | delete objects *
----
* 0 | exit *
****
enter your selection : 2
****
* 1 | modify a person instance *
----
* 2 | modify a city instance *
----
* 3 | modify a country instance *
----
* 4 | modify a client instance *
----
* 5 | modify a client_employee instance *
----
* 6 | modify an employee instance *
----
* 7 | modify a hotel instance *
----
* 8 | modify a reservation for hotel *
----
* 9 | modify a monument instance *
----
* 10 | modify a museum instance *
----
* 11 | modify a reservation instance *
----
* 12 | modify a place to go instance *
----
* 13 | modify a theater instance *

```
* 14 | modify a tour *
*-----------------------------------------------*
* 15 | modify a tour reservation *
*************************************************

 enter your selection :4
 enter the social_security number of the client :035-60-7738
************************************
* 1 | social security number *
*---------------------------------*
* 2 | last name *
*---------------------------------*
* 3 | first name *
*---------------------------------*
* 4 | birthdate *
*---------------------------------*
* 5 | address *
*---------------------------------*
* 6 | reservations *
*---------------------------------*
* 7 | place of stay *
*---------------------------------*
* 8 | departure date *
*---------------------------------*
* 9 | returning date *
*---------------------------------*
* 10| number of persons *
*---------------------------------*
* 11| bill *
************************************
 selection : 6
************************
* 1 | insert *
*----------------------*
* 2 | delete *
************************
 selection : 2
 enter the reservation you want to delete(or insert) :
 enter the client :
 enter the social_security number :
035-60-7738
 the client already exsist in the database
 enter the reservation date :
 enter the week :Monday
```

```
enter the month :June
enter the date :22
enter the year :1992
****************************
* 1 | create new object *
*-------------------------*
* 2 | modify data *
*-------------------------*
* 3 | retrieve data *
*-------------------------*
* 4 | delete objects *
*-------------------------*
* 0 | exit *
****************************
enter your selection : 3
**************************************************
* 1 | get all the person instance *
*-----------------------------------------------*
* 2 | get all the city instance *
*-----------------------------------------------*
* 3 | get all the country instance *
*-----------------------------------------------*
* 4 | get all the client instance *
*-----------------------------------------------*
* 5 | get all the client_employee instance *
*-----------------------------------------------*
* 6 | get all the employee instance *
*-----------------------------------------------*
* 7 | get all the hotel instance *
*-----------------------------------------------*
* 8 | get all the reservation for hotel *
*-----------------------------------------------*
* 9 | get all the monument instance *
*-----------------------------------------------*
* 10 | get all the museum instance *
*-----------------------------------------------*
* 11 | get all the reservation instance *
*-----------------------------------------------*
* 12 | get all the place to go instance *
*-----------------------------------------------*
* 13 | get all the theater instance *
*-----------------------------------------------*
* 14 | get all the tour *
*-----------------------------------------------*
```

```
* 15 | get all the tour reservation *
************************************************

 enter your selection :4
NAME          SC#    BIRTHDATEADDRESS
**************************************************************************
**************
Li , Yu-Fang035-60-7738April 11 , 19679 Alumni Ave.
                        Providence,02906
                        USA

SC#    # OF PERSONSPLACE TO STAYDEP DATERETURN DATE BILL
**************************************************************************
**************
035-60-773820New YorkMay 18 , 1992 MondayMay 25 , 1992 Monday500.000000
 display the reservations for some client ?(yes or no)yes
 enter the social security # of the client :035-60-7738
 reservation information for this person :
SC# of CLIENT RESERVATION DATE NUMBER OF PERSON
**************************************************************************
****
035-60-7738May 19 , 19922
035-60-7738May 20 , 19923
035-60-7738May 21 , 19924
035-60-7738May 22 , 19925
 enter the other client ?(yes or no)no
*****************************
* 1 | create new object *
*-------------------------*
* 2 | modify data *
*-------------------------*
* 3 | retrieve data *
*-------------------------*
* 4 | delete objects *
*-------------------------*
* 0 | exit *
*****************************
 enter your selection : 2
*****************************************************
* 1 | modify a person instance *
*-----------------------------------------------*
* 2 | modify a city instance *
*-----------------------------------------------*
* 3 | modify a country instance *
```

```
*-----------------------------------------------*
* 4 | modify a client instance *
*-----------------------------------------------*
* 5 | modify a client_employee instance *
*-----------------------------------------------*
* 6 | modify an employee instance *
*-----------------------------------------------*
* 7 | modify a hotel instance *
*-----------------------------------------------*
* 8 | modify a reservation for hotel *
*-----------------------------------------------*
* 9 | modify a monument instance *
*-----------------------------------------------*
* 10 | modify a museum instance *
*-----------------------------------------------*
* 11 | modify a reservation instance *
*-----------------------------------------------*
* 12 | modify a place to go instance *
*-----------------------------------------------*
* 13 | modify a theater instance *
*-----------------------------------------------*
* 14 | modify a tour *
*-----------------------------------------------*
* 15 | modify a tour reservation *
*************************************************

 enter your selection :4
 enter the social_security number of the client :035-60-7738
**********************************
* 1 | social security number *
*--------------------------------*
* 2 | last name *
*--------------------------------*
* 3 | first name *
*--------------------------------*
* 4 | birthdate *
*--------------------------------*
* 5 | address *
*--------------------------------*
* 6 | reservations *
*--------------------------------*
* 7 | place of stay *
*--------------------------------*
* 8 | departure date *
```

```
*--------------------------------*
* 9 | returning date *
*--------------------------------*
* 10| number of persons *
*--------------------------------*
* 11| bill *
************************************
 selection : 10
 enter the new number of persons : 9
****************************
* 1 | create new object *
*-------------------------*
* 2 | modify data *
*-------------------------*
* 3 | retrieve data *
*-------------------------*
* 4 | delete objects *
*-------------------------*
* 0 | exit *
****************************
 enter your selection : 3
************************************************************
* 1 | get all the person instance *
*-----------------------------------------------*
* 2 | get all the city instance *
*-----------------------------------------------*
* 3 | get all the country instance *
*-----------------------------------------------*
* 4 | get all the client instance *
*-----------------------------------------------*
* 5 | get all the client_employee instance *
*-----------------------------------------------*
* 6 | get all the employee instance *
*-----------------------------------------------*
* 7 | get all the hotel instance *
*-----------------------------------------------*
* 8 | get all the reservation for hotel *
*-----------------------------------------------*
* 9 | get all the monument instance *
*-----------------------------------------------*
* 10 | get all the museum instance *
*-----------------------------------------------*
* 11 | get all the reservation instance *
*-----------------------------------------------*
```

```
* 12 | get all the place to go instance *
*-----------------------------------------------*
* 13 | get all the theater instance *
*-----------------------------------------------*
* 14 | get all the tour *
*-----------------------------------------------*
* 15 | get all the tour reservation *
*************************************************

 enter your selection :4
NAME          SC#    BIRTHDATEADDRESS
*************************************************************************
**************
Li , Yu-Fang035-60-7738April 11 , 19679 Alumni Ave.
                      Providence,02906
                      USA

SC#    # OF PERSONSPLACE TO STAYDEP DATERETURN DATE BILL
*************************************************************************
**************
035-60-77389 New YorkMay 18 , 1992 MondayMay 25 , 1992 Monday500.000000
 display the reservations for some client ?(yes or no)yes
 enter the social security # of the client :035-60-7738
 reservation information for this person :
SC# of CLIENT RESERVATION DATE NUMBER OF PERSON
*************************************************************************
****
035-60-7738May 19 , 19922
035-60-7738May 20 , 19923
035-60-7738May 21 , 19924
035-60-7738May 22 , 19925
 enter the other client ?(yes or no)no
*****************************
* 1 | create new object *
*-------------------------*
* 2 | modify data *
*-------------------------*
* 3 | retrieve data *
*-------------------------*
* 4 | delete objects *
*-------------------------*
* 0 | exit *
*****************************
 enter your selection : 2
```

```
*****************************************************
* 1 | modify a person instance *
*-----------------------------------------------*
* 2 | modify a city instance *
*-----------------------------------------------*
* 3 | modify a country instance *
*-----------------------------------------------*
* 4 | modify a client instance *
*-----------------------------------------------*
* 5 | modify a client_employee instance *
*-----------------------------------------------*
* 6 | modify an employee instance *
*-----------------------------------------------*
* 7 | modify a hotel instance *
*-----------------------------------------------*
* 8 | modify a reservation for hotel *
*-----------------------------------------------*
* 9 | modify a monument instance *
*-----------------------------------------------*
* 10 | modify a museum instance *
*-----------------------------------------------*
* 11 | modify a reservation instance *
*-----------------------------------------------*
* 12 | modify a place to go instance *
*-----------------------------------------------*
* 13 | modify a theater instance *
*-----------------------------------------------*
* 14 | modify a tour *
*-----------------------------------------------*
* 15 | modify a tour reservation *
*****************************************************

 enter your selection :4
 enter the social_security number of the client :035-60-7738
***********************************
* 1 | social security number *
*---------------------------------*
* 2 | last name *
*---------------------------------*
* 3 | first name *
*---------------------------------*
* 4 | birthdate *
*---------------------------------*
* 5 | address *
```

```
*--------------------------------*
* 6 | reservations *
*--------------------------------*
* 7 | place of stay *
*--------------------------------*
* 8 | departure date *
*--------------------------------*
* 9 | returning date *
*--------------------------------*
* 10| number of persons *
*--------------------------------*
* 11| bill *
**************************************
 selection : 6
************************
* 1 | insert *
*---------------------*
* 2 | delete *
************************
 selection : 1
 enter the reservation you want to delete(or insert) :
 enter the client :
 enter the social_security number :
035-60-7738
 the client already exsist in the database
 enter the reservation date :
 enter the week :Tuesday
 enter the month :June
 enter the date :23
 enter the year :1992
 enter the number of persons :
40
the inserted reservation's number of persons > client's number of persons!
 please change it!
****************************
* 1 | create new object *
*-------------------------*
* 2 | modify data *
*-------------------------*
* 3 | retrieve data *
*-------------------------*
* 4 | delete objects *
*-------------------------*
* 0 | exit *
```

```
****************************
 enter your selection : 2
*****************************************************
* 1 | modify a person instance *
*-----------------------------------------------*
* 2 | modify a city instance *
*-----------------------------------------------*
* 3 | modify a country instance *
*-----------------------------------------------*
* 4 | modify a client instance *
*-----------------------------------------------*
* 5 | modify a client_employee instance *
*-----------------------------------------------*
* 6 | modify an employee instance *
*-----------------------------------------------*
* 7 | modify a hotel instance *
*-----------------------------------------------*
* 8 | modify a reservation for hotel *
*-----------------------------------------------*
* 9 | modify a monument instance *
*-----------------------------------------------*
* 10 | modify a museum instance *
*-----------------------------------------------*
* 11 | modify a reservation instance *
*-----------------------------------------------*
* 12 | modify a place to go instance *
*-----------------------------------------------*
* 13 | modify a theater instance *
*-----------------------------------------------*
* 14 | modify a tour *
*-----------------------------------------------*
* 15 | modify a tour reservation *
*****************************************************

 enter your selection :11
 enter the client's social security number : 035-60-7738
 enter the date of reservation:
 enter the week :Tuesday
 enter the month :June
 enter the date :23
 enter the year :1992
**********************************
* 1 | client *
*-------------------------------*
```

```
* 2 | date of reservation *
*------------------------------*
* 3 | number of persons *
**********************************
3
 enter the new number of persons : 8
****************************
* 1 | create new object *
*-------------------------*
* 2 | modify data *
*-------------------------*
* 3 | retrieve data *
*-------------------------*
* 4 | delete objects *
*-------------------------*
* 0 | exit *
****************************
 enter your selection : 3
*****************************************************
* 1 | get all the person instance *
*-----------------------------------------------*
* 2 | get all the city instance *
*-----------------------------------------------*
* 3 | get all the country instance *
*-----------------------------------------------*
* 4 | get all the client instance *
*-----------------------------------------------*
* 5 | get all the client_employee instance *
*-----------------------------------------------*
* 6 | get all the employee instance *
*-----------------------------------------------*
* 7 | get all the hotel instance *
*-----------------------------------------------*
* 8 | get all the reservation for hotel *
*-----------------------------------------------*
* 9 | get all the monument instance *
*-----------------------------------------------*
* 10 | get all the museum instance *
*-----------------------------------------------*
* 11 | get all the reservation instance *
*-----------------------------------------------*
* 12 | get all the place to go instance *
*-----------------------------------------------*
* 13 | get all the theater instance *
```

```
*-----------------------------------------------*
* 14 | get all the tour *
*-----------------------------------------------*
* 15 | get all the tour reservation *
*************************************************

 enter your selection :4
NAME          SC#    BIRTHDATEADDRESS
*****************************************************************************
**************
Li , Yu-Fang035-60-7738April 11 , 19679 Alumni Ave.
                      Providence,02906
                      USA

SC#    # OF PERSONSPLACE TO STAYDEP DATERETURN DATE BILL
*****************************************************************************
**************
035-60-77389 New YorkMay 18 , 1992 MondayMay 25 , 1992 Monday500.000000
 display the reservations for some client ?(yes or no)035-60-7738
*****************************
* 1 | create new object *
*-------------------------*
* 2 | modify data *
*-------------------------*
* 3 | retrieve data *
*-------------------------*
* 4 | delete objects *
*-------------------------*
* 0 | exit *
*****************************
 enter your selection : 3
*************************************************
* 1 | get all the person instance *
*-----------------------------------------------*
* 2 | get all the city instance *
*-----------------------------------------------*
* 3 | get all the country instance *
*-----------------------------------------------*
* 4 | get all the client instance *
*-----------------------------------------------*
* 5 | get all the client_employee instance *
*-----------------------------------------------*
* 6 | get all the employee instance *
*-----------------------------------------------*
```

```
* 7 | get all the hotel instance *
*-----------------------------------------------*
* 8 | get all the reservation for hotel *
*-----------------------------------------------*
* 9 | get all the monument instance *
*-----------------------------------------------*
* 10 | get all the museum instance *
*-----------------------------------------------*
* 11 | get all the reservation instance *
*-----------------------------------------------*
* 12 | get all the place to go instance *
*-----------------------------------------------*
* 13 | get all the theater instance *
*-----------------------------------------------*
* 14 | get all the tour *
*-----------------------------------------------*
* 15 | get all the tour reservation *
************************************************

 enter your selection :4
NAME          SC#    BIRTHDATEADDRESS
********************************************************************************
**************
Li , Yu-Fang035-60-7738April 11 , 19679 Alumni Ave.
                      Providence,02906
                      USA

SC#    # OF PERSONSPLACE TO STAYDEP DATERETURN DATE BILL
********************************************************************************
**************
035-60-77389 New YorkMay 18 , 1992 MondayMay 25 , 1992 Monday500.000000
 display the reservations for some client ?(yes or no)yes
 enter the social security # of the client :035-60-7738
 reservation information for this person :
SC# of CLIENT RESERVATION DATE NUMBER OF PERSON
********************************************************************************
****
035-60-7738May 21 , 19924
035-60-7738May 22 , 19925
035-60-7738May 20 , 19923
035-60-7738May 19 , 19922
035-60-7738June 23 , 19928
 enter the other client ?(yes or no)no
****************************
```

```
* 1 | create new object *
*-------------------------*
* 2 | modify data *
*-------------------------*
* 3 | retrieve data *
*-------------------------*
* 4 | delete objects *
*-------------------------*
* 0 | exit *
****************************
 enter your selection : 0
(dulcimer)2--> exit


exit

cript done on Mon Jun 22 15:18:16 1992
```

Chapter 5: Future Work & Conclusions

From the analysis of chapter 3, we can see that this system provides an efficient ways to maintain the constraint integrity and data consistency. In traditional relational database system, sometimes you have to check a lot of data though some of them have no connection with this constraint. On the contrary, the strategy presents in this system only check those objects that are really affected by the imposed constraint. Theoretically, the algorithms used in this system is an efficient way to impose constraints onto the database since we don't have to waste time to check those objects that wouldn't be affected by the constraint. And, realistically, from the time and space analysis, it's clear that this strategy didn't take too much time and space to achieve our goal.

The pitfall here is that the constraint type we can deal with is still limited. This is partly because of the built-in operations in ObjectStore and partly because of the object-equality problem in the object-oriented database realm. The other drawback is that the user has to write some part of the system on their own because of the difficulties in ObjectStore as we discussed in section 2.3.1. Therefore, to solve all of these problems, we have to overcome the difficulties in ObjectStore first. After that, we'll go back to find a way to deal with the object-equality problem so that this system can manage a lot more kinds of constraint.

Acknowledgment

I gratefully appreciate the assistance of Ted Leung, who provided me all the algorithms to build this system and gave me a lot of advise during these months.

Appendix A: Code Listing

A.1 Constraint compiler & demon generator

```
/********** lex file parser.l**************/
%{
#define sym_eof 0
#define sym_key_end 1
#define sym_key_integer 2
#define sym_key_real 3
#define sym_key_constraint 4
#define sym_key_string 5
#define sym_key_class 6
#define sym_op_idiv 7
#define sym_op_and 8
#define sym_op_or 9
#define sym_op_mod 10
#define sym_identifier 11
#define sym_float 12
#define sym_integer 13
#define sym_op_plus 14
#define sym_op_minus 15
#define sym_op_mult 16
#define sym_op_fdiv 17
#define sym_op_lesseq 18
#define sym_op_greateq 19
#define sym_op_less 20
#define sym_op_great 21
#define sym_op_noteq 22
#define sym_op_eq 23
#define sym_op_in 24
#define sym_op_must_in 25
#define sym_asgn 26
#define sym_comma 27
#define sym_period 28
#define sym_semi 29
#define sym_colon 30
#define sym_r_parn 31
#define sym_l_parn 32
#define sym_r_bracket 33
#define sym_l_bracket 34
#define sym_error 35
#define sym_l_sbracket 36
#define sym_r_sbracket 37
#define sym_key_set 38
#define sym_op_not_in 39
#define sym_key_forEvery 40
```

```
double atof();
int line_number=1;
%}

alphabet ([A-Za-z]|\_)
digit [0-9]
exp (E|e)[-+]?{digit}+

%%
[\t];
\n {line_number++;}
END | end {return (sym_key_end);}
INT | int {return (sym_key_integer);}
FLOAT | float {return(sym_key_real);}
DIV | div {return (sym_op_idiv);}
MOD | mod {return(sym_op_mod);}
CLASS | class {return(sym_key_class);}
CONSTRAINT | constraint {return(sym_key_constraint);}
IN | in {return(sym_op_in);}
NOT_IN | not_in {return(sym_op_not_in);}
MUST_IN | must_in {return(sym_op_must_in);}
STRING | string {return(sym_key_string);}
SET | set {return(sym_key_set);}
forEvery {return(sym_key_forEvery);}

{alphabet}({alphabet}|{digit})* |
{alphabet}+\-({alphabet} | {digit})* |
\"{alphabet}+\-({alphabet} | {digit})*\" |
\"{alphabet}({alphabet}|{digit})*\" {return(sym_identifier);}
{digit}+\.{digit}+{exp} |
{digit}+\.{digit}+ |
{digit} + {exp} {return(sym_float);}
{digit}+ {return(sym_integer);}

\&\& {return (sym_op_and);}
\N {return(sym_op_or);}
\+ {return(sym_op_plus);}
\- {return(sym_op_minus);}
\* {return(sym_op_mult);}
\/ {return(sym_op_fdiv);}
\<\= {return(sym_op_lesseq);}
\>\= {return(sym_op_greateq);}
\< {return(sym_op_less);}
\> {return(sym_op_great);}
```

```
\!\= {return(sym_op_noteq);}
\=\= {return(sym_op_eq);}
\= {return(sym_asgn);}
\, {return(sym_comma);}
\. {return(sym_period);}
\; {return(sym_semi);}
\: {return(sym_colon);}
\) {return(sym_r_parn);}
\({return(sym_l_parn);}
\} {return(sym_r_bracket);}
\{{return(sym_l_bracket);}
\[{return(sym_l_sbracket);}
\] {return(sym_r_sbracket);}
. {return(sym_error);}

%%

/************ yacc file parser.y ***************************/
%{
#include <stdio.h>
#include <string.h>
#include "typedef.h"

extern MEM_NODE *member_node(), *link_mem_node();
extern NODE *creat_node(), *link();
extern SYM_TABLE *creat_sym_table(), *link_sym_table();
extern char *type_check(), *type_of_var(), *subtype_of_class();
extern void acap_push(), init_asgn_node(),acap_free(),print_acap();
extern void compute_acap(), print_set_acap();
extern char *acap_subtype_of_class();
extern void acap_asgn_free();
extern void template_generate(),print_template_code();

NODE *type_table = NULL;
SYM_TABLE *sym_table = NULL;
char *ConstraintName = NULL;
char *constraint_op = NULL;
char *ckattr;

%}
%union {
 MEM_NODE *mem_val;
 NODE *node_val;
 SYM_TABLE *sym_val;
```

```
char *str_val;
}
%token sym_eof 0
%token sym_key_end 1
%token sym_key_integer 2
%token sym_key_real 3
%token sym_key_constraint 4
%token sym_key_string 5
%token sym_key_class 6
%token sym_op_idiv 7
%token sym_op_and 8
%token sym_op_or 9
%token sym_op_mod 10
%token sym_identifier 11
%token sym_float 12
%token sym_integer 13
%token sym_op_plus 14
%token sym_op_minus 15
%token sym_op_mult 16
%token sym_op_fdiv 17
%token sym_op_lesseq 18
%token sym_op_greateq 19
%token sym_op_less 20
%token sym_op_great 21
%token sym_op_noteq 22
%token sym_op_eq 23
%token sym_op_in 24
%token sym_op_must_in 25
%token sym_asgn 26
%token sym_comma 27
%token sym_period 28
%token sym_semi 29
%token sym_colon 30
%token sym_r_parn 31
%token sym_l_parn 32
%token sym_r_bracket 33
%token sym_l_bracket 34
%token sym_error 35
%token sym_l_sbracket 36
%token sym_r_sbracket 37
%token sym_key_set 38
%token sym_op_not_in 39
%token sym_key_forEvery 40
```

```
%type <node_val> opt_decl_specifiers
%type <node_val> opt_decl_specifier
%type <node_val> class_specifier
%type <str_val> class_head
%type <mem_val> opt_member_list
%type <str_val> decl_identifier
%type <str_val> identifier
%type <mem_val> opt_member
%type <str_val> type
%type <str_val> std_type
%type <str_val> set_type
%type <str_val> set
%type <sym_val> const_decl
%type <sym_val> const_decls
%type <str_val> const_statement
%type <str_val> expression
%type <str_val> simple_expression
%type <str_val> term
%type <str_val> factor
%type <str_val> num

%left sym_op_and sym_op_or
%left sym_op_idiv sym_op_mod
%left sym_op_plus sym_op_minus
%left sym_op_mult sym_op_fdiv
%%
program: prog_decls
 | error
;
prog_decls: declaration
 | prog_decls declaration
 | error
;
declaration: opt_decl_specifiers
 {type_table = $1;}
 constraints
 | error
;
opt_decl_specifiers: {$$= NULL;}
 | opt_decl_specifier
 {$$= $1;}
 | opt_decl_specifiers opt_decl_specifier
 {$$= link ($1, $2);}
;
```

```
opt_decl_specifier: class_specifier
 {$$= $1;}
;
class_specifier: class_head sym_l_bracket opt_member_list sym_r_bracket
 {$$= creat_node ($1, $3);}
;
class_head: sym_key_class decl_identifier
 {$$= $2;
        }
;
opt_member_list: opt_member
 {$$= $1;}
 | opt_member_list opt_member
 {$$= link_mem_node ($1, $2);}
;
opt_member: decl_identifier sym_colon type sym_semi
 {$$= member_node ($1, $3);
        }
;
type: std_type
 {$$= $1;}
 | class_head
 {$$= $1;}
 | set_type
 {$$= $1;}
;
set_type: set sym_l_sbracket decl_identifier sym_r_sbracket
 {$$= strcat($1, $3);
        }
;
set: sym_key_set
 {$$= (char *) malloc (strlen("*")+1);
        strcpy ($$, "*");}
;
constraints: constraint
 | constraints constraint
;
constraint: sym_key_constraint sym_identifier
 {
         ConstraintName = (char *) malloc(strlen(yytext)+1);
         strcpy(ConstraintName,yytext);
        }
 const_decls
 {printf("*********************************************\n");
```

```
printf("SYMBLE TABLE *\n");
        printf("*****************************************\n");
        print_sym_table($4);
printf ("\n");
        sym_table = $4;
        }
const_constraint sym_key_end
{
        template_generate(ConstraintName,constraint_op,ckattr);
print_set_acap();
        printf ("\n\n");
        /*print_template_code(ConstraintName);*/
        free(ConstraintName);
        free_sym_table($4);
        acap_free();
        acap_asgn_free();
        }

;
const_decls: const_decl
 {$$= $1;}
 | const_decls const_decl
 {$$= link_sym_table ($1, $2);}
;
const_decl: type decl_identifier sym_semi
 {$$= creat_sym_table($1, $2);
        }
;
const_constraint: const_expression
 | const_statements const_expression{/*print_acap();*/}
 | error
;
const_expression: expression sym_semi {compute_acap();}
 | forEvery_expression sym_l_bracket const_constraint sym_r_bracket
 | error
;
const_statements:
 | const_statement
 | const_statements const_statement
 | error
;
const_statement: identifier assignop
 {acap_push("=");}
 expression sym_semi
```

```
{$$= type_check ($1, $4);
                    init_asgn_node();}
| forEvery_expression sym_l_bracket const_statements sym_r_bracket
{$$= NULL;}
;
forEvery_expression: sym_key_forEvery expression {acap_free();}
;

expression: simple_expression
 {$$= $1;}
 | simple_expression relop
 {acap_push("==");}
 simple_expression
 {$$= type_check ($1, $4);
          if ($$!= NULL)
                    {$$= (char *) malloc(strlen("int")+1);
                    strcpy($$, "int");
                    }
          }
;
simple_expression: term
 {$$= $1;}
 | sign term
 {$$= $2;}
 | simple_expression addop term
 {$$= type_check ($1, $3);
                    acap_free();}
;
term: factor
 {$$= $1;}
 | term mulop factor
 {$$= $1;
 acap_free();}
;
factor: identifier
 {$$= $1;
          }
 | num
 {$$= $1;}
 | sym_l_parn expression sym_r_parn
 {$$= $2;
          acap_free();}
;
sign: sym_op_plus
```

```
 | sym_op_minus
 | error
;
num: sym_integer
 {$$= (char *) malloc (strlen("int")+1);
 strcpy($$, "int");}
 | sym_float
 {$$= (char *) malloc (strlen("float")+1);
        strcpy($$,"float ");}
;
relop: sym_op_greateq
 {if (constraint_op != NULL) free(constraint_op) ;
        constraint_op = (char *) malloc (strlen(yytext)+1) ;
        strcpy(constraint_op,yytext);
        }
 | sym_op_great
 {if (constraint_op != NULL) free(constraint_op);
        constraint_op = (char *) malloc (strlen(yytext)+1);
        strcpy(constraint_op,yytext);
        }
 | sym_op_lesseq
 {if (constraint_op!= NULL) free(constraint_op);
        constraint_op = (char *) malloc (strlen(yytext)+1);
        strcpy(constraint_op,yytext);
        }
 | sym_op_less
 {if (constraint_op!= NULL) free(constraint_op);
        constraint_op = (char *) malloc (strlen(yytext)+1);
        strcpy(constraint_op,yytext);
        }
 | sym_op_eq
 {if (constraint_op!= NULL) free(constraint_op);
        constraint_op = (char *) malloc (strlen(yytext)+1);
        strcpy(constraint_op,yytext);
        }
 | sym_op_noteq
 {if (constraint_op!= NULL) free(constraint_op);
        constraint_op = (char *) malloc (strlen(yytext)+1);
        strcpy(constraint_op,yytext);
        }
 | sym_op_must_in
 {if (constraint_op!= NULL) free(constraint_op);
        constraint_op = (char *) malloc (strlen("==")+1);
        strcpy (constraint_op,"==");
```

```
        }
;
assignop: sym_asgn
 | sym_op_in
 | sym_op_not_in
;
addop: sym_op_plus
 | sym_op_minus
 | sym_op_or
;
mulop: sym_op_idiv
 | sym_op_mod
 | sym_op_mult
 | sym_op_fdiv
 | sym_op_and
;
std_type: sym_key_string
 {$$= (char *) malloc(strlen(yytext)+1);
        strcpy($$, yytext);
        }
 | sym_key_real
 {$$= (char *) malloc(strlen(yytext)+1);
        strcpy($$, yytext);
        }
 | sym_key_integer
 {$$= (char *) malloc(strlen(yytext)+1);
        strcpy($$, yytext);
        }
;
decl_identifier: sym_identifier
 {$$= (char *) malloc (strlen(yytext)+1);
        strcpy ($$, yytext);
                free(ckattr);
                ckattr = (char *) malloc (strlen(yytext)+1);
        strcpy (ckattr, yytext);
        }
;
identifier: decl_identifier
 {$$= type_of_var($1);
        acap_push($1);
        }
 | identifier sym_period decl_identifier
 {$$= subtype_of_class($1,$3);
        acap_push(acap_subtype_of_class($1,$3));
```

```
        }
;
%%

#include "lex.yy.c"



/********** generating a constraint instance declaration ***********/
char *const_inst_generator(const,type,fp,l_ad,l_ap,r_ad,r_ap)
char *type , *const ;
FILE *fp ;
struct type_node *l_ad,*l_ap,*r_ad,*r_ap ;
{
 static char *ci;
 int l_cache_set = False , r_cache_set = False ;

 if (contains_set(l_ad) == True || contains_set(l_ap) == True ||
 ((contains_set(l_ap) == Empty && contains_set(l_ad) == False) &&
 (contains_set(r_ap) == Empty && contains_set(r_ad) == False)))
 l_cache_set = True ;

 if (contains_set(r_ad) == True || contains_set(r_ap) == True ||
 ((contains_set(l_ap) == Empty && contains_set(l_ad) == False) &&
 (contains_set(r_ap) == Empty && contains_set(r_ad) == False)))
 r_cache_set = True ;

 if(l_cache_set == False && r_cache_set == False ) {
 ci = (char *) malloc(strlen("C_1<")+1);
 strcpy(ci,"C_1<");
 strcat(ci,type);
 strcat(ci,"> *");
 strcat(ci,const);
 return(ci);
 }
 if(l_cache_set == False && r_cache_set == True ) {
 ci = (char *) malloc(strlen("C_2<")+1);
 strcpy(ci,"C_2<");
 strcat(ci,type);
 strcat(ci,"> *");
 strcat(ci,const);
 return(ci);
```

```
}
if(l_cache_set == True && r_cache_set == False ) {
ci = (char *) malloc(strlen("C_3<")+1);
strcpy(ci,"C_3<");
strcat(ci,type);
strcat(ci,"> *");
strcat(ci,const);
return(ci);
}
if(l_cache_set == True && r_cache_set == True ) {
ci = (char *) malloc(strlen("C_4<")+1);
strcpy(ci,"C_4<");
strcat(ci,type);
strcat(ci,"> *");
strcat(ci,const);
return(ci);
}
}
```

/* generate procedures for a class that is in AD set and is a set type */

```
generate_adset_code(side ,ckattr,const,set,ap_set,fp,op,ci)
char *set,*op ,*const,*ckattr,*ap_set,*ci;
FILE *fp ;
int side ;
{
 NODE *class_type = find_class_type(ap_set) ;
 MEM_NODE *subtype ;

 if (class_type == NULL) return ;
 subtype = class_type->member ;
 while(subtype!=NULL && strcmp(subtype->ty_name, set))
 subtype = subtype->next ;


 fputs ("void ",fp) ;
 fputs(ap_set,fp) ;
 fputs("_",fp);
 fputs(subtype->mem_name+1,fp);
 fputs("(",fp);
 if (strchr(subtype->ty_name,'*')!=NULL)
 fputs(subtype->ty_name+1,fp);
 else fputs(subtype->ty_name,fp);
 fputs(" *element, int mod_status,",fp);
 fputs(ci,fp);
```

```
fputs(“){\n”,fp) ;

fputs(“\t\tif (mod_status == INSERT) {\n”,fp) ;
fputs(“\t\t demonize(element,”,fp);
fputs(const,fp);
fputs(“);\n”,fp) ;
fputs(“\t\t “,fp) ;
fputs(const,fp);
fputs(“->add_ckval”,fp) ;
switch(side)
{
case l :
fputs(“1(“,fp) ;
break ;
case r :
fputs(“2(“,fp) ;
break ;
default : break ;
}
fputs(“element->”,fp) ;
fputs(ckattr+1,fp);
fputs(“());\n\t}\n”,fp) ;

fputs(“\t\tif (mod_status == DELETE) {\n”,fp) ;
fputs(“\t\t undemonize(element,”,fp);
fputs(const,fp);
fputs(“);\n”,fp) ;
fputs(“\t\t “,fp) ;
fputs(const,fp);
fputs(“->del_ckval”,fp) ;
switch(side)
{
case l :
fputs(“1(“,fp) ;
break ;
case r :
fputs(“2(“,fp) ;
break ;
default : break ;
}
fputs(“element->”,fp) ;
fputs(ckattr+1,fp);
fputs(“());\n\t}\n}\n”,fp) ;
```

```
/* fputs("\t\tif updated(element)\n",fp);
 fputs("\t\t if ",fp) ;
 switch(side)
 {
 case 1 :
 fputs(const,fp) ;
 fputs("->ckval2)\n",fp);
 fputs(not(op),fp) ;
 fputs("(element->",fp) ;
 fputs(ckattr,fp) ;
 break ;
 case 2 :
 fputs("(",fp) ;
 fputs(const,fp) ;
 fputs("->ckval1",fp) ;
 fputs(op,fp) ;
 fputs("element->",fp) ;
 fputs(ckattr,fp) ;
 fputs(")\n",fp) ;
 default: break ;
 }
 fputs("\t\t\t ",fp) ;
 fputs(const,fp);
 fputs("->new_ckval",fp) ;
 switch(side)
 {
 case 1 :
 fputs("1(",fp) ;
 break ;
 case r :
 fputs("2(",fp) ;
 break ;
 default : break ;
 }
 fputs("element->",fp) ;
 fputs(ckattr,fp);
 fputs(") ;\n",fp) ;
 fputs("\t\t else\n\t\t\t exception();\n",fp) ;
 fputs("\t}\n",fp) ; */

}
```

/* generating procedures for a class that is in AD set but is not a set type */

```
generate_ad_code(side,ckattr,const,set,fp,op,ci)
```

```
char *set,*op,*const,*ckattr ,*ci;
FILE *fp ;
int side ;
{ fputs("\n",fp) ;
 fputs("void ",fp) ;
 fputs(set,fp) ;
 fputs("_",fp) ;
 fputs(ckattr+1,fp) ;
 fputs("(int ",fp);
 fputs(ckattr,fp);
 fputs(", int new_",fp);
 fputs(ckattr,fp);
 fputs(", ",fp);
 fputs(ci,fp);
 fputs("){\n" , fp) ;

 fputs("\tif (",fp) ;
 switch(side)
 {
 case 1 :
 fputs("new_",fp) ;
 fputs(ckattr,fp) ;
 fputs(op,fp) ;
 fputs(const,fp) ;
 fputs("->ckval2",fp) ;
 break ;

 case 2 :
 fputs("new_",fp);
 fputs(ckattr,fp) ;
 fputs(not(op),fp) ;
 fputs(const,fp) ;
 fputs("->ckval1",fp) ;
 break ;
 default : break ;
 }
 fputs(")\n\t\t",fp) ;
 fputs(const,fp) ;
 fputs("->new_ckval",fp);
 switch(side)
 {
 case 1:
 fputs("1",fp) ;
 break ;
```

```
case 2 :
fputs("2",fp) ;
break ;
default : break ;
}
fputs("(",fp) ;
fputs(ckattr,fp) ;
fputs(",new_",fp) ;
fputs(ckattr,fp) ;
fputs(") ;\n",fp) ;

fputs("\telse\n",fp) ;
fputs("\t\texception() ;\n}\n",fp) ;

}
```

/* generating procedures for a class that is in AP set but is not a set type */

```
generate_ap_code(side,const,set,next,fp,ci)
char *const ;
char *set ,*ci;
struct type_node *next ;
FILE *fp ;
int side ;
{
 NODE *class_type = find_class_type(set) ;
 MEM_NODE *subtype ;

 if (class_type == NULL) return ;
 subtype = class_type->member ;
 while(subtype!=NULL)
 {
 if (!strcmp(subtype->ty_name, next->set))
        { fputs("\n",fp) ;
         fputs("void ",fp) ;
         fputs(set,fp) ;
         fputs("_" , fp) ;
         fputs(subtype->mem_name+1,fp) ;
         fputs("(",fp);
         if (strchr(subtype->ty_name,'*')!=NULL)
         fputs(subtype->ty_name+1,fp);
         else fputs(subtype->ty_name,fp);
         fputs(" *",fp);
         fputs(subtype->mem_name,fp);
         fputs(", ",fp);
```

```
        if (strchr(subtype->ty_name,'*')!=NULL)
        fputs(subtype->ty_name+1,fp);
        else fputs(subtype->ty_name,fp);
        fputs(" *new_",fp);
        fputs(subtype->mem_name,fp);
        fputs("){\n",fp) ;
        fputs("\t\tSplicePath(",fp) ;
        fputs(subtype->mem_name,fp) ;
        fputs(",new_",fp) ;
        fputs(subtype->mem_name,fp) ;
        fputs(");\n}",fp) ;fputs("\n",fp);
        }
subtype = subtype->next ;
}
}
```

/* generating procedures for a class that is in AP set and is also a set type */

```
generate_apset_code(side,const,set,ap_set,fp,ci)
char *set,*const,*ap_set,*ci;
FILE *fp ;
int side ;
{
NODE *class_type = find_class_type(ap_set) ;
MEM_NODE *subtype ;

if (class_type == NULL) return ;
subtype = class_type->member ;
while(subtype!=NULL && strcmp(subtype->ty_name, set))
subtype = subtype->next ;


fputs ("void ",fp) ;
fputs(ap_set,fp) ;
fputs("_",fp);
fputs(subtype->mem_name+1,fp);
fputs("(",fp);
if (strchr(subtype->ty_name,'*')!=NULL)
fputs(subtype->ty_name+1,fp);
else fputs(subtype->ty_name,fp);
fputs(" *element, int mod_status, ",fp);
fputs(ci,fp);
fputs("){\n",fp) ;

fputs("\tif (mod_status == INSERT) \n",fp) ;
```

```
fputs("\t demonize(element, ",fp);
fputs(const,fp);
fputs(");\n",fp) ;

fputs("\tif (mod_status == DELETE) \n",fp) ;
fputs("\t undemonize(element, ",fp);
fputs(const,fp);
fputs(");\n }\n",fp) ;

}
```

/* generating procedures for all the classes in AD */

```
code_for_ad(side,const,ad,ap,fp,op,ckattr,ci)
struct type_node *ad ,*ap;
FILE *fp ;
int side ;
char *const , *op,*ckattr,*ci ;
{
 struct type_node *temp = ad ;
 struct type_node *temp1 = ap ;

 if (temp1!=NULL)
 while( temp1->next !=NULL) temp1 = temp1->next ;

 while ( temp != NULL )
 {
 if (is_a_set(temp) == True )
        {
         generate_adset_code(side,ckattr ,const,temp->set,temp1->set,fp,op,ci) ;
         generate_ad_code(side,ckattr,const,temp->set+1,fp,op,ci) ;
        }
 else generate_ad_code(side,ckattr,const,temp->set,fp,op,ci) ;
 temp = temp->next ;
 }
}
```

/* generating procedures for all the classes in AP */

```
code_for_ap(side,const,ap,ad ,fp,ci)
struct type_node *ap , *ad ;
FILE *fp ;
char *const,*ci ;
int side ;
{
 struct type_node *temp = ap , *temp1 = NULL ;
```

```
struct type_node *next = NULL ;

while (temp!=NULL)
{
if (temp->next == NULL) next = ad ;
else next = temp->next ;

if (is_a_set(temp) == True )
        {
         generate_apset_code(side,const,temp->set,temp1->set,fp,ci) ;
         generate_ap_code(side,const,temp->set+1,next,fp),ci ;
        }
else generate_ap_code(side,const,temp->set,next,fp,ci) ;
temp1 = temp ;
temp = temp->next ;
 }
}
```

/* generating procedures for the constraint */

```
template_code_generator(const,fp , l_ad,l_ap,r_ad,r_ap,op,ckattr,ci)
FILE *fp ;
struct type_node *l_ad,*l_ap,*r_ad,*r_ap ;
char *op ,*const,*ckattr,*ci;
{
 code_for_ad(l,const,l_ad , l_ap,fp,op,ckattr,ci) ;
 code_for_ad(r,const,r_ad,r_ap,fp,op,ckattr,ci) ;
 code_for_ap(l,const,l_ap,l_ad,fp,ci) ;
 code_for_ap(r,const,r_ap,r_ad,fp,ci) ;

}
```

/* generating constraint declaration and procedures */

```
template_generate(ConstraintName,op,ckattr)
char *ConstraintName;
char *op,*ckattr ;
{
 FILE *fp ;
 char *type ,*ci;
 fp = fopen(ConstraintName,"w") ;
 type = type_of_var(ckattr) ;
 header(fp);
 ci = const_inst_generator(ConstraintName,type,fp,l_ad,l_ap,r_ad,r_ap) ;
 template_code_generator(ConstraintName,fp , l_ad,l_ap,r_ad,r_ap,op,ckattr,ci) ;
 fclose(fp) ;
```

```
}
```

A.2 Database schema

```
#include <ostore/ostore.hh>
#include <ostore/coll.hh>
#include "String.h"
#define INSERT 1
#define DELETE 2
#define NULL_DEMON 0

template <class Type> class C_1;
template <class Type> class C_2;
template <class Type> class C_3;
template <class Type> class C_4;

/************* CLASS Person AND ITS SUB-CLASSES ***********/
class Address ;
class Date ;
extern void Person_social_security(char*);
extern void Person_name_demon(string*);
extern void Person_first_name(string*);
extern void Person_birthdate(Date*);
extern void Person_address(Address*);

class Person {
 public :
        Person(char *social_security = NULL , string *name = NULL ,
                string *first_name = NULL ,
         Date *birthdate = NULL , Address *address = NULL) ;
        ~Person() ;
        void print() ;
        Person(database *db = 0 ,Person *peron = NULL ) ;
        char *social_security() { return _social_security ; }
        void social_security(char *new__social_security) {
         if(social_security_demon_ptr_list!=NULL_DEMON)
                Person_social_security(new__social_security) ;
         delete _social_security ;
         _social_security = new__social_security ;
 }
        string *name() { return _name ; } ;
        void name(string *new__name) {
         if(name_demon_ptr_list!=NULL_DEMON)
                Person_name_demon(new__name) ;
```

```
        delete _name ;
        _name = new__name ; }
       string *first_name() { return _first_name ; } ;
       void first_name(string *new__first_name ) {
        if(first_name_demon_ptr_list!=NULL_DEMON)
                Person_first_name(new__first_name) ;
        delete _first_name ;
        _first_name = new__first_name ; }
       Date *birthdate() { return _birthdate ; } ;
       void birthdate(Date *new__birthdate) {
if(birthdate_demon_ptr_list!=NULL_DEMON)
                Person_birthdate(new__birthdate) ;
        _birthdate = new__birthdate ; }
       Address *address() { return _address ; } ;
       void address(Address *new__address) {
if(address_demon_ptr_list!=NULL_DEMON)
                Person_address(new__address) ;
        _address = new__address ; }

       int social_security_demon_ptr_list;
       int name_demon_ptr_list;
       int first_name_demon_ptr_list;
       int birthdate_demon_ptr_list;
       int address_demon_ptr_list;

protected :
       char *_social_security ;
       string *_name ;
       string *_first_name ;
       Date *_birthdate ;
       Address *_address ;
} ;

class Country ;
class City ;
extern void Address_number(int);
extern void Address_street(string*);
extern void Address_zip_code(string*);
extern void Address_country(Country*);
extern void Address_city(City*);

class Address {
 public :
       Address(Country *country = NULL , string *street = NULL ,
```

```
        City *city = NULL , string *zip_code = NULL ,
        int number = 0 ) ;
        Address(Address *addr = NULL ) ;
~Address() ;
        int number() { return _number ; }
        void number(int new__number ) {
        if( number_demon_ptr_list!=NULL_DEMON)
                Address_number(new__number) ;
        _number = new__number ; }
        string *street() { return _street ; } ;
        void street(string *new__street) {
if(street_demon_ptr_list!=NULL_DEMON)
                Address_street(new__street) ;
        _street = new__street ; }
        City *city() { return _city ; } ;
        void city(City *new__city) {
if(city_demon_ptr_list!=NULL_DEMON)
                Address_city(new__city) ;
        _city = new__city ; }
        string *zip_code() { return _zip_code ; } ;
        void zip_code(string *new__zip_code) {
if(zip_code_demon_ptr_list!=NULL_DEMON)
                Address_zip_code(new__zip_code) ;
        _zip_code = new__zip_code ; } ;
        Country *country() { return _country ; } ;
        void country(Country *new__country) {
if(country_demon_ptr_list!=NULL_DEMON)
                Address_country(new__country) ;
        _country = new__country ; } ;

        C_1<int> *number_c1;
        C_2<int> *number_c2;
        C_3<int> *number_c3;
        C_4<int> *number_c4;
        int number_demon_ptr_list;
        int street_demon_ptr_list;
        int zip_code_demon_ptr_list;
        int country_demon_ptr_list;
        int city_demon_ptr_list;


private :
        int _number ;
        string *_street ;
```

```
        City *_city ;
        string *_zip_code;
        Country * _country ;
} ;
extern void Money_money(float);
class Money {
 public :
        Money(float m = 0) ;
        ~Money() ;
        float money() { return _money ; } ;
        void money(float new__money) {
 if(money_demon_ptr_list!=NULL_DEMON)
                Money_money(new__money) ;
         _money = new__money ; }

        int money_demon_ptr_list;

 private :
        float _money ;
} ;

class City ;
class Date ;
class Reservation ;
class Address ;
extern void Client_reservations(Reservation*,int,C_2<int>*);
extern void Client_number_of_persons(int,int,C_2<int>*);
extern void Client_place_of_stay(City *) ;
extern void Client_departure_date(Date *) ;
extern void Client_returning_date(Date *) ;
extern void Client_bill(float) ;
class Client : public virtual Person {
 public :
        Client( City *place_of_stay = NULL , Date *dep_date = NULL ,
                Date *return_date = NULL, int num_persons=0 , float bill=0 ,
                char *social_security = NULL , string *name = NULL ,
                string *first_name = NULL ,
                 Date *birthdate = NULL , Address *address = NULL ) ;
        ~Client() ;
        os_List<Reservation*> _reservations ;
        os_List<Reservation*> reservations(void *) { return _reservations ; } ;
        void reservations(int mod_status = 0 ,Reservation *element= NULL) {
         if (reservations_demon_ptr_list!=NULL_DEMON)
                 Client_reservations(element,mod_status,number_of_persons_c2) ;
```

```
 if (mod_status == INSERT) _reservations.insert(element) ;
 if (mod_status == DELETE) _reservations.remove(element) ; }
City *place_of_stay() { return _place_of_stay ; } ;
void place_of_stay(City *new__place_of_stay) {
 if(place_of_stay_demon_ptr_list!=NULL_DEMON)
        Client_place_of_stay(new__place_of_stay) ;
 _place_of_stay = new__place_of_stay ; }
Date *departure_date() { return _departure_date ; } ;
void departure_date(Date *new__departure_date) {
 if(departure_date_demon_ptr_list!=NULL_DEMON)
        Client_departure_date(new__departure_date) ;
 _departure_date = new__departure_date ; }
Date *returning_date() { return _returning_date ; } ;
void returning_date(Date *new__returning_date) {
 if(returning_date_demon_ptr_list!=NULL_DEMON)
        Client_returning_date(new__returning_date) ;
 _returning_date = new__returning_date ; }
int number_of_persons() { return _number_of_persons ; }
void number_of_persons(int new__number_of_persons) {
 if (number_of_persons_demon_ptr_list!=NULL_DEMON)
 Client_number_of_persons(_number_of_persons,
                new__number_of_persons,number_of_persons_c2) ;
 _number_of_persons = new__number_of_persons ; }
float bill() { return _bill ; }
void bill(float new__bill ) {
 if(bill_demon_ptr_list!=NULL_DEMON)
        Client_bill(new__bill) ;
 _bill = new__bill ; }

C_1<int> *number_of_persons_c1;
C_2<int> *number_of_persons_c2;
C_3<int> *number_of_persons_c3;
C_4<int> *number_of_persons_c4;
int reservations_demon_ptr_list;
int number_of_persons_demon_ptr_list;
int place_of_stay_demon_ptr_list;
int departure_date_demon_ptr_list;
int returning_date_demon_ptr_list;
int bill_demon_ptr_list;


protected :
        City *_place_of_stay ;
        Date *_departure_date ;
```

```
        Date *_returning_date ;
        int _number_of_persons ;
        float _bill ;
} ;

extern void Employee_seniority(int) ;
extern void Employee_title(string*) ;
extern void Employee_salary(Money*) ;

class Employee : public virtual Person {
 public :
        Employee(int sen= 0 , string *title = NULL , Money *s = NULL ,
                char *social_security = NULL ,
                string *name = NULL, string *first_name = NULL,
                Date *birthdate = NULL , Address *addres = NULL );
        ~Employee() ;
        int seniority() { return _seniority ; } ;
        void seniority(int new__seniority) {
         if(seniority_demon_ptr_list!=NULL_DEMON)
                Employee_seniority(new__seniority) ;
         _seniority = new__seniority ; }
        string *title() { return _title ; } ;
        void title(string *new__title) {
         if(title_demon_ptr_list!=NULL_DEMON)
                Employee_title(new__title) ;
         _title = new__title ; }
        Money *salary() { return _salary ; } ;
        void salary(Money *new__salary) {
         if(salary_demon_ptr_list!=NULL_DEMON)
                Employee_salary(new__salary) ;
         _salary = new__salary ; }

        C_1<int> *seniority_c1;
        C_2<int> *seniority_c2;
        C_3<int> *seniority_c3;
        C_4<int> *seniority_c4;
        int seniority_demon_ptr_list;
        int title_demon_ptr_list;
        int salary_demon_ptr_list;


 protected :
        int _seniority ;
        string *_title ;
```

```
        Money *_salary ;
        } ;
extern void Client_employee_discount(float);

class Client_employee : public Client , public Employee
 {
 public :
        Client_employee(float discount = 0 ,int sen = 0 , Money *s = NULL,
                        string *title = NULL , City *place_of_stay= NULL ,
                        Date *dep_date= NULL , Date *return_date = NULL ,
                        int num_persons = 0 , float bill = 0 ,
                        char *social_security = NULL,
                        string *name = NULL , string *first_name = NULL ,
                        Date *birthdate = NULL , Address *address = NULL ) ;
        ~Client_employee() ;
        float discount() { return _discount ; } ;
        void discount(float new__discount) {
         if(discount_demon_ptr_list!=NULL_DEMON)
                Client_employee_discount(new__discount) ;
         _discount = new__discount ; }

        int discount_demon_ptr_list;

 private :
        float _discount ;

} ;

extern void Reservation_number_of_persons(int,int,C_2<int>*);
extern void Reservation_client(Client*) ;
extern void Reservation_date_of_reservation(Date*) ;
/************** CLASS Reservation AND ITS SUB-CLASSES ******/

class Reservation {
 public :
        Reservation(Client * client = NULL ,Date *date = NULL , int = 0) ;
        ~Reservation() ;
        Client *client() { return _client ; } ;
        void client(Client *new__client) {
         if(client_demon_ptr_list!=NULL_DEMON)
                Reservation_client(new__client) ;
         _client = new__client ; }
        Date *date_of_reservation() { return _date_of_reservation; };
        void date_of_reservation(Date *new__date_of_reservation) {
```

```
        if(date_of_reservation_demon_ptr_list!=NULL_DEMON)
                Reservation_date_of_reservation(new__date_of_reservation) ;
        _date_of_reservation = new__date_of_reservation ; }
       int number_of_persons() { return _number_of_persons ; } ;
       void number_of_persons(int new__number_of_persons) {
        if (number_of_persons_demon_ptr_list!=NULL_DEMON)
        Reservation_number_of_persons(_number_of_persons,
                        new__number_of_persons,number_of_persons_c2) ;
        _number_of_persons = new__number_of_persons; }

       C_1<int> *number_of_persons_c1;
       C_2<int> *number_of_persons_c2;
       C_3<int> *number_of_persons_c3;
       C_4<int> *number_of_persons_c4;
       int number_of_persons_demon_ptr_list ;
       int client_demon_ptr_list;
       int date_of_reservation_demon_ptr_list;


protected :
       Client *_client ;
       Date *_date_of_reservation ;
       int _number_of_persons ;
       } ;

class Hotel ;
class Client ;
extern void Hotel_reservation_end_date(Date*) ;
extern void Hotel_reservation_selected_hotel(Hotel*) ;
extern void Hotel_reservation_type_of_stay(string*) ;
class Hotel_reservation : public Reservation
 {
 public :
       Hotel_reservation(Date* end_date= NULL, Hotel* selected_hotel = NULL ,
                string *type_of_stay = NULL , Client *c = NULL ,
                Date* date_of_reservation = NULL , int num_of_persons = 0 ) ;
       ~Hotel_reservation() ;
       Date *end_date() { return _end_date ; } ;
       void end_date(Date *new__end_date) {
        if(end_date_demon_ptr_list!=NULL_DEMON)
                Hotel_reservation_end_date(new__end_date) ;
        _end_date = new__end_date ; }
       Hotel *selected_hotel() { return _selected_hotel ; };
       void selected_hotel(Hotel* new__selected_hotel) {
```

```
        if(selected_hotel_demon_ptr_list!=NULL_DEMON)
                Hotel_reservation_selected_hotel(new__selected_hotel) ;
        _selected_hotel = new__selected_hotel ; }
        string *type_of_stay() { return _type_of_stay ; } ;
        void type_of_stay(string *new__type_of_stay) {
        if(type_of_stay_demon_ptr_list!=NULL_DEMON)
                Hotel_reservation_type_of_stay(new__type_of_stay) ;
_type_of_stay = new__type_of_stay ; }

        int end_date_demon_ptr_list;
        int selected_hotel_demon_ptr_list;
        int type_of_stay_demon_ptr_list;


private:
        Date *_end_date ;
        Hotel *_selected_hotel ;
        string *_type_of_stay ;
} ;

class Tour ;
class Client ;
class Date ;
extern void Tour_reservation_selected_tour(Tour*);

class Tour_reservation : public Reservation {
 public :
        Tour_reservation(Tour *selected_tour= NULL , Client *c = NULL ,
         Date* date_of_reservation = NULL , int num_of_persons = 0) ;
        ~Tour_reservation() ;
        Tour *selected_tour() { return _selected_tour ; } ;
        void selected_tour(Tour *new__selected_tour) {
         if(selected_tour_demon_ptr_list!=NULL_DEMON)
                Tour_reservation_selected_tour(new__selected_tour) ;
         _selected_tour = new__selected_tour ; }

        int selected_tour_demon_ptr_list;

 private :
        Tour *_selected_tour ;
} ;


/****************** CLASS Hotel ***************/
```

```
class Day_price ;
extern void Hotel_name(string*);
extern void Hotel_address(Address*) ;
extern void Hotel_stars(int) ;
extern void Hotel_facilities(string* ) ;
extern void Hotel_day_price(Day_price *) ;
class Hotel {
 public :
        Hotel(string *name = NULL , Address *address = NULL ,
                int stars = 0, Day_price *day_price = NULL ) ;
        ~Hotel() ;
        string *name() { return _name ; } ;
        void name(string *new__name) {
         if(name_demon_ptr_list!=NULL_DEMON)
                Hotel_name(new__name) ;
         delete _name ;
         _name = new__name ; }
        Address *address() { return _address ; } ;
        void address(Address *new__address) {
         if(address_demon_ptr_list!=NULL_DEMON)
                Hotel_address(new__address) ;
         delete _address ;
         _address = new__address ; }
        int stars() { return _stars ; }
        void stars(int new__stars) {
         if(stars_demon_ptr_list!=NULL_DEMON)
                Hotel_stars(new__stars) ;
         _stars = new__stars ; }
        os_Set<string*> _facilities ;
        void facilities(int mod_status= 0 , string *element=NULL) {
         if(facilities_demon_ptr_list!=NULL_DEMON)
                Hotel_facilities(element) ;
         if (mod_status == INSERT) _facilities.insert(element) ;
         if (mod_status == DELETE) _facilities.remove(element) ;
         }
        Day_price *day_price() { return _day_price ; } ;
        void day_price(Day_price *new__day_price) {
         if(day_price_demon_ptr_list!=NULL_DEMON)
                Hotel_day_price(new__day_price) ;
         _day_price = new__day_price ; }

        C_1<int> *stars_c1;
        C_2<int> *stars_c2;
```

```
        C_3<int> *stars_c3;
        C_4<int> *stars_c4;
        int name_demon_ptr_list;
        int address_demon_ptr_list;
        int stars_demon_ptr_list;
        int facilities_demon_ptr_list;
        int day_price_demon_ptr_list;


protected :
        string *_name ;
        Address *_address ;
        int _stars ;
//      os_Set<string*> _facilities ;
        Day_price *_day_price ;
} ;
extern void Day_price_breakfast_price(Money *) ;
extern void Day_price_half_board_price(Money *) ;
extern void Day_price_full_board_price(Money *) ;
class Day_price {
 public :
        Day_price(Money *bp , Money *hbp , Money *fbp) ;
        ~Day_price() ;
        Money *breakfast_price() { return _breakfast_price ; };
        void breakfast_price(Money * new__breakfast_price) {
         if(breakfast_price_demon_ptr_list!=NULL_DEMON)
                Day_price_breakfast_price(new__breakfast_price) ;
         _breakfast_price = new__breakfast_price ; }
        Money *half_board_price() { return _half_board_price ; } ;
        void half_board_price(Money *new__half_board_price ) {
         if(half_board_price_demon_ptr_list!=NULL_DEMON)
                Day_price_half_board_price(new__half_board_price ) ;
         _half_board_price = new__half_board_price ; }
        Money *full_board_price() { return _full_board_price ; } ;
        void full_board_price(Money *new__full_board_price ) {
         if(full_board_price_demon_ptr_list!=NULL_DEMON)
                Day_price_full_board_price(new__full_board_price) ;
         _full_board_price = new__full_board_price ; }
        int breakfast_price_demon_ptr_list;
        int half_board_price_demon_ptr_list;
        int full_board_price_demon_ptr_list;

 private :
        Money *_breakfast_price ;
```

```
        Money *_half_board_price ;
        Money *_full_board_price ;
} ;

extern void Place_to_go_name(string *) ;
extern void Place_to_go_address(Address *) ;
extern void Place_to_go_price(Money *) ;

/***************** CLASS Place_to_go AND ITS SUB-CLASSES ***/

class Place_to_go {
 public :
        Place_to_go(string *name = NULL , Address *addr = NULL ,
                        Money *price = NULL) ;
        ~Place_to_go() ;
        string *name() { return _name ; } ;
        void name(string *new__name) {
         if(name_demon_ptr_list!=NULL_DEMON)
                Place_to_go_name(new__name) ;
         _name = new__name ; }
        Address *address() { return _address ; };
        void address(Address *new__address) {
         if(address_demon_ptr_list!=NULL_DEMON)
                Place_to_go_address(new__address) ;
         _address = new__address ; }
        Money *price() { return _price ;} ;
        void price(Money *new__price) {
         if(price_demon_ptr_list!=NULL_DEMON)
                Place_to_go_price(new__price) ;
         _price = new__price ; }

        int name_demon_ptr_list;
        int address_demon_ptr_list;
        int price_demon_ptr_list;


 protected :
        string *_name ;
        Money *_price ;
        Address *_address ;
} ;

extern void Monument_building_date(Date *) ;
extern void Monument_architect(Person *) ;
```

```
class Monument : public Place_to_go {
 public :
        Monument(Date *build_date = NULL , Person *architect= NULL ,
                string *name = NULL , Address *addr = NULL ,
                Money *price = NULL ) ;
        ~Monument() ;
        Date *building_date() { return _building_date ; } ;
        void building_date(Date *new__building_date ) {
         if(building_date_demon_ptr_list!=NULL_DEMON)
                Monument_building_date(new__building_date) ;
         _building_date = new__building_date; }
        Person *architect() { return _architect ;}
        void architect(Person *new__architect ) {
         if(architect_demon_ptr_list!=NULL_DEMON)
                Monument_architect(new__architect) ;
         _architect = new__architect ; }

        int building_date_demon_ptr_list;
        int architect_demon_ptr_list;


 private :
        Date *_building_date ;
        Person *_architect ;
} ;

extern void Museum_topic(string *) ;
extern void Museum_artists(Person* ) ;

class Museum : public Place_to_go {
 public :
        Museum(string *topic , string *name , Address *addr ,
                Money *price = NULL ) ;
        ~Museum() ;
        string *topic() { return _topic ; } ;
        void topic(string *new__topic) {
 if(topic_demon_ptr_list!=NULL_DEMON)
                Museum_topic(new__topic) ;
 _topic = new__topic ; }
 os_Set<Person*> _artists ;
 void artists(int mod_status = 0 , Person *element= NULL) {
         if(artists_demon_ptr_list!=NULL_DEMON)
                Museum_artists(element) ;
         if (mod_status == INSERT) _artists.insert(element) ;
```

```
        if (mod_status == DELETE) _artists.remove(element) ;
        }
        int topic_demon_ptr_list;
        int artists_demon_ptr_list;

private :
        string *_topic ;
//      os_Set<Person*> *_artists ;
} ;

extern void Theater_building_date(Date *) ;
extern void Theater_number_of_seats(int) ;
class Theater : public Place_to_go {
public :
        Theater(Date *building_date , int num_of_seats ,
string *name , Address *addr,Money *price ) ;
        ~Theater() ;
        Date *building_date() { return _building_date ; } ;
        void building_date(Date *new__building_date) {
         if(building_date_demon_ptr_list!=NULL_DEMON)
                Theater_building_date(new__building_date) ;
         _building_date = new__building_date ; }
        int number_of_seats() { return _number_of_seats ; } ;
        void number_of_seats(int new__number_of_seats) {
         if(number_of_seats_demon_ptr_list!=NULL_DEMON)
                Theater_number_of_seats(new__number_of_seats) ;
         _number_of_seats = new__number_of_seats ; }

        C_1<int> *number_of_seats_c1;
        C_2<int> *number_of_seats_c2;
        C_3<int> *number_of_seats_c3;
        C_4<int> *number_of_seats_c4;
        int building_date_demon_ptr_list;
        int number_of_seats_demon_ptr_list;


private :
        Date *_building_date ;
        int _number_of_seats ;
} ;

/******************** CLASS Country *******/
extern void Country_name(string *) ;
extern void Country_cities(City * ) ;
```

```
class Country {
 public :
        Country(string *name = NULL) ;
        ~Country() ;
        void print() ;
        string *name() { return _name ; } ;
        void name(string *new__name) {
         if(name_demon_ptr_list!=NULL_DEMON)
                Country_name(new__name) ;
         _name = new__name ; }
        os_Set<City *> cities() { return _cities ; }
        void cities(int mod_status = 0 , City *element = NULL){
         if(cities_demon_ptr_list!=NULL_DEMON)
                Country_cities(element) ;
         if (mod_status == INSERT) _cities.insert(element) ;
 if (mod_status == DELETE) _cities.remove(element) ;
         }
        os_Set<City*> _cities ;

        int name_demon_ptr_list;
        int cities_demon_ptr_list;


 private :
        string *_name ;
        //os_Set<City*> _cities ;
} ;

extern void City_name(string *) ;
extern void City_country(Country *) ;
extern void City_places_to_go(Place_to_go * ) ;
extern void City_hotels(Hotel * ) ;
extern void City_tours(Tour*) ;
class City {
 public :
        City(string *name = NULL , Country *country = NULL ) ;
        ~City() ;
        string *name() { return _name ; } ;
        void print() ;
        void name(string *new__name) {
         if(name_demon_ptr_list!=NULL_DEMON)
                City_name(new__name) ;
         _name = new__name ; }
        Country *country() { return _country ; } ;
```

```
        void country(Country *new__country) {
         if(country_demon_ptr_list!=NULL_DEMON)
                City_country(new__country) ;
         _country = new__country ; }
        os_Set<Place_to_go *> _places_to_go ;
        void places_to_go(int mod_status = 0 , Place_to_go *element = NULL) {
         if(places_to_go_demon_ptr_list!=NULL_DEMON)
                City_places_to_go(element) ;
         if ( mod_status == INSERT ) _places_to_go.insert(element) ;
         if (mod_status == DELETE ) _places_to_go.remove(element) ; }
        os_Set<Hotel *> _hotels ;
        void hotels(int mod_status = 0 ,Hotel *element = NULL) {
         if(hotels_demon_ptr_list!=NULL_DEMON)
                City_hotels(element) ;
         if ( mod_status == INSERT ) _hotels.insert(element) ;
         if (mod_status == DELETE ) _hotels.remove(element) ;
         }
        os_List<Tour*> _tours ;
        void tours(int mod_status = 0 , Tour *element = NULL) {
         if(tours_demon_ptr_list!=NULL_DEMON)
                City_tours(element) ;
         if ( mod_status == INSERT ) _tours.insert(element) ;
         if (mod_status == DELETE ) _tours.remove(element) ;
        }

        int name_demon_ptr_list;
        int country_demon_ptr_list;
        int places_to_go_demon_ptr_list;
        int hotels_demon_ptr_list;
        int tours_demon_ptr_list;

private :
        string *_name ;
        Country *_country ;
//      os_Set<Place_to_go*> *_places_to_go ;
//      os_Set<Hotel*> *_hotels ;
//      os_List<Tour*> *_tours ;

} ;


/******************** CLASS Tour ****/
extern  void Stage_what(Place_to_go *) ;
extern  void Stage_time(string *) ;
```

```
class Stage {
 public :
        Stage(Place_to_go * , string *) ;
        ~Stage() ;
        Place_to_go *what() { return _what ; } ;
        void what(Place_to_go *new__what) {
         if(what_demon_ptr_list!=NULL_DEMON)
                Stage_what(new__what) ;
         _what = new__what ; }
        string *time() { return _time ; } ;
        void time(string *new__time) {
         if(time_demon_ptr_list!=NULL_DEMON)
                Stage_time(new__time) ;
         _time = new__time ; }

        int what_demon_ptr_list;
        int time_demon_ptr_list;


 private :
        Place_to_go *_what ;
        string *_time ;
} ;
extern  void Tour_capacity(int) ;
extern  void Tour_availability(int);
extern  void Tour_theme(string *);
extern  void Tour_city(City*);
extern  void Tour_description(Stage*);
extern  void Tour_price(Money*);
class Tour {
 public :
        Tour(string *theme = NULL , City *city = NULL , int capa = 0 ,
         int avail= 0 , Money *price =NULL) ;
        ~Tour() ;
        int capacity() { return _capacity ; } ;
        void capacity(int new__capacity) {
         if(capacity_demon_ptr_list!=NULL_DEMON)
                Tour_capacity(new__capacity) ;
         _capacity = new__capacity ; }
        int availability() { return _availability ; } ;
        void availability(int new__availability ) {
         if(availability_demon_ptr_list!=NULL_DEMON)
                Tour_availability(new__availability) ;
         _availability = new__availability ;}
```

```
        string *theme() { return _theme; }
        void theme(string *new__theme) {
         if(theme_demon_ptr_list!=NULL_DEMON)
                Tour_theme(new__theme) ;
         _theme = new__theme ; }
        City *city() { return _city ; } ;
        void city(City *new__city) {
         if(city_demon_ptr_list!=NULL_DEMON)
                Tour_city(new__city) ;
         _city = new__city ; }
        os_List<Stage*> _description ;
        void description(int mod_status = 0 ,Stage *element = NULL) {
         if(description_demon_ptr_list!=NULL_DEMON)
                Tour_description(element) ;
         if (mod_status == INSERT) _description.insert(element) ;
         if (mod_status == DELETE) _description.remove(element) ; }
        Money *price() { return _price ; } ;
        void price(Money *new__price) {
         if(price_demon_ptr_list!=NULL_DEMON)
                Tour_price(new__price) ;
         _price = new__price ; }

        C_1<int> *capacity_c1;
        C_2<int> *capacity_c2;
        C_3<int> *capacity_c3;
        C_4<int> *capacity_c4;
        C_1<int> *availability_c1;
        C_2<int> *availability_c2;
        C_3<int> *availability_c3;
        C_4<int> *availability_c4;
        int capacity_demon_ptr_list;
        int availability_demon_ptr_list;
        int theme_demon_ptr_list;
        int city_demon_ptr_list;
        int description_demon_ptr_list;
        int price_demon_ptr_list;


private :
        int _capacity ;
        int _availability ;
        string *_theme ;
        City *_city ;
//      os_List<Stage*> _description ;
```

```
        Money *_price ;
} ;

extern void Date_month(string *) ;
extern  void Date_date(int) ;
extern  void Date_year(int) ;
extern  void Date_week(string *) ;
class Date {
 public :
        Date(string *mon = NULL , int date = 0 , int year = 0 ,
         string *week = NULL) ;
        ~Date() ;
        void print() ;
        string *month() { return _month ; }
        void month(string *new__month) {
         if(month_demon_ptr_list!=NULL_DEMON)
                Date_month(new__month) ;
         _month = new__month ; }
        int date() { return _date ; }
        void date(int new__date) {
         if(date_demon_ptr_list!=NULL_DEMON)
                Date_date(new__date) ;
         _date = new__date ; }
        int year() { return _year ; }
        void year(int new__year ) {
         if(year_demon_ptr_list!=NULL_DEMON)
                Date_year(new__year) ;
         _year = new__year ; }
        string *week() { return _week ; }
        void week(string *new__week) {
         if(week_demon_ptr_list!=NULL_DEMON)
                Date_week(new__week) ;
         _week = new__week ; }

        C_1<int> *year_c1;
        C_2<int> *year_c2;
        C_3<int> *year_c3;
        C_4<int> *year_c4;
        C_1<int> *date_c1;
        C_2<int> *date_c2;
        C_3<int> *date_c3;
        C_4<int> *date_c4;
        int month_demon_ptr_list;
        int date_demon_ptr_list;
```

```
        int year_demon_ptr_list;
        int week_demon_ptr_list;

private :

        string *_month ;
        int _date ;
        int _year ;
        string *_week ;
} ;

template <class Type>
class C_1 {
 public :
 C_1(Type t1=0 ,Type t2=0) { ckval1 = t1 ; ckval2 = t2 ;};
 ~C_1() {};
 void new_ckval1(Type old_a , Type a) { ckval1 = a ; Type b = old_a;}
 void new_ckval2(Type old_a , Type a) { ckval2 = a ; Type b = old_a;}

 Type ckval1 ;
 Type ckval2 ;
} ;


template <class Type>
class C_2 {
 public :
 C_2() {
 ckval1 = 0;
 ckval2 = os_Set<Type *>();
 };
 ~C_2() { foreach(Type *i, ckval2)
                delete i;
         };
 void new_ckval1(Type old_a , Type a) { ckval1 = a ; Type b=old_a;} ;
 void new_ckval2(Type old_a , Type a) {
 foreach(Type *elm,ckval2) {
        if (*elm == old_a) {
         ckval2.remove(elm) ;
         Type *temp = new(db) Type;
         *temp = a;
         ckval2.insert(temp) ;
         return ;
         }
```

```
};
}
void add_ckval2(Type a) {
        Type *i = new(db) Type ;
        *i = a ;
        ckval2.insert(i) ; } ;
void del_ckval2(Type a) {
        foreach(Type *i,ckval2)
         if (*i==a) {
                 ckval2.remove(i) ;
                  return ;
                 }
        } ;


Type ckval1 ;
os_Set<Type *> ckval2 ;
} ;

template <class Type>
class C_3 {
public :
C_3() { ckval2 = 0 ; ckval1 = os_Set<Type*>() ; };
~C_3() { foreach(Type *i,ckval1)
                 delete i;
         };
void new_ckval1(Type old_a , Type a) {
foreach(Type *elm,ckval1)
        if (*elm == old_a) {
         ckval1.remove(elm) ;
         Type *temp = new Type;
         *temp = a ;
         ckval1.insert(temp) ;
         return ;
        }
};
void add_ckval1(Type a) {
        Type *i = new(db) Type ;
        *i = a ;
        ckval1.insert(i) ;
        } ;
void del_ckval1(Type a) {
        foreach(Type *i,ckval1)
         if (*i==a) {
```

```
                ckval1.remove(i) ;
                return;
                }
        } ;
void new_ckval2(Type old_a ,Type a) { ckval2 = a ; Type b=old_a;} ;

os_Set<Type *> ckval1 ;
int ckval2 ;
} ;



template <class Type>
class C_4 {
 public :
 C_4() { ckval1 = os_Set<Type*>() ; ckval2 = os_Set<Type*>() ;};
 ~C_4() { foreach(Type *i,ckval1) delete i;
        foreach(Type *j,ckval2) delete j; } ;
 void new_ckval1(Type old_a , Type a) {
 foreach(Type *elm,ckval1) {
        if (*elm == old_a) {
         ckval1.remove(elm) ;
         Type *temp = new Type;
         *temp = a;
         ckval1.insert(temp) ;
         return ;
         }
 }
 };
 void add_ckval1(Type a) {
        Type *i = new(db) Type ;
        *i = a ;
        ckval1.insert(i) ;
        } ;
 void del_ckval1(Type a) {
        foreach(Type *i,ckval2) {
         if (*i==a) ckval2.remove(i) ;
         return;}
         } ;
 void new_ckval2(Type old_a , Type a) {
        foreach(Type *elm,ckval2) {
         if (*elm == old_a) {
                ckval2.remove(elm) ;
         Type *temp = new Type;
```

```
        *temp = a ;
                ckval2.insert(temp) ;
                return ;
        }
        }
};
void add_ckval2(Type a) {
        Type *i = new(db) Type ;
        *i = a ;
        ckval2.insert(i) ;
        } ;
void del_ckval2(Type a) {
        foreach(Type *i,ckval2)
        if (*i==a) {
                ckval2.remove(i) ;
                return;
                }
        } ;

os_Set<Type*> ckval1 ;
os_Set<Type*> ckval2 ;
} ;

extern database *db ;
persistent <db> os_Set<Person *> people ;
persistent <db> os_Set<City *> city_set ;
persistent <db> os_Set<Country *> country_set ;
persistent<db> os_Set<Client *> client_set ;
persistent<db> os_Set<Client_employee *> client_employee_set ;
persistent<db> os_Set<Employee *> employee_set ;
persistent<db> os_Set<Hotel *> hotel_set ;
persistent<db> os_Set<Hotel_reservation *> hotel_reservation_set ;
persistent<db> os_Set<Monument *> monument_set ;
persistent<db> os_Set<Museum *> museum_set ;
persistent<db> os_Set<Reservation *> reservation_set ;
persistent<db> os_Set<Place_to_go *> place_to_go_set ;
persistent<db> os_Set<Theater *> theater_set ;
persistent<db> os_Set<Tour *> tour_set ;
persistent<db> os_Set<Tour_reservation *> tour_reservation_set ;
```

A.3 Demon Attach

```
/************ FILE demonize.cc ************************/
#include <ostore/ostore.hh>
```

```
#include <ostore/coll.hh>
#include <iostream.h>
#define TRUE 1
#define FALSE 0
#define NULL_DEMON 0
#define ATT_DEMON 1
#include "schema.hh"

extern void Client_number_of_persons(int _number_of_persons,
                        int new__number_of_persons,C_2<int>*);
extern void Client_reservations(Reservation *element,int mod_status,C_2<int>*);
extern void Reservation_number_of_persons(int _number_of_persons,
                        int new__number_of_persons,C_2<int>*);
extern void null_demon(char *);
extern void null_demon (Address *);
extern void null_demon (Country *) ;
extern void null_demon (Date *) ;
extern void null_demon (Tour *) ;
extern void null_demon (City *) ;
extern void null_demon (Reservation *) ;
extern void null_demon (Place_to_go *) ;
extern void null_demon (Stage *);
extern void null_demon (Hotel *) ;
extern void null_demon (Money*);
extern void null_demon (Day_price *) ;
extern void null_demon (string *) ;
extern void null_demon (Person *) ;
extern void null_demon (Client *);
extern void null_demon (float) ;
extern void null_demon (int) ;
extern void null_demon (int,int) ;
extern void null_demon (Reservation*,int, C_2<int>*) ;

database *db;


void main(int, char **argv)
{
 db = database ::open(argv[1]);

 do_transaction() {
 void *v = NULL;
 foreach(Client *cl, client_set) {
 C_2<int> *C1 = new(db) C_2<int>;
```

```
C1->ckval1 = cl->number_of_persons();
printf("ckval1=%i\n",C1->ckval1);
cl->number_of_persons_demon_ptr_list = ATT_DEMON;
cl->reservations_demon_ptr_list = ATT_DEMON;
cl->number_of_persons_c2 = C1 ;
foreach( Reservation *r, cl->reservations(v)) {
int *temp = new(db) int;
        *temp = r->number_of_persons() ;
        C1->ckval2.insert(temp);
        r->number_of_persons_demon_ptr_list = ATT_DEMON;
        r->number_of_persons_c2 = C1;
}
foreach(int *i, C1->ckval2) printf("ckval2=%i\n",*i);
}
}

db->close();
}


/************** FILE undemonize.cc ***********************************/
#include <ostore/ostore.hh>
#include <ostore/coll.hh>
#include <iostream.h>
#define TRUE 1
#define FALSE 0
#define NULL_DEMON 0
#include "schema.hh"

extern void Client_number_of_persons(int _number_of_persons,
                        int new__number_of_persons,C_2<int>*);
extern void Client_reservations(Reservation *element,int mod_status,C_2<int>*);
extern void Reservation_number_of_persons(int _number_of_persons,
                        int new__number_of_persons,C_2<int>*);
extern void null_demon(char *);
extern void null_demon (Address *);
extern void null_demon (Country *) ;
extern void null_demon (Date *) ;
extern void null_demon (Tour *) ;
extern void null_demon (City *) ;
extern void null_demon (Reservation *) ;
extern void null_demon (Place_to_go *) ;
extern void null_demon (Stage *);
extern void null_demon (Hotel *) ;
```

```
extern void null_demon (Money*);
extern void null_demon (Day_price *) ;
extern void null_demon (string *) ;
extern void null_demon (Person *) ;
extern void null_demon (Client *);
extern void null_demon (float) ;
extern void null_demon (int) ;
extern void null_demon (int,int,C_2<int>*) ;
extern void null_demon (Reservation*,int, C_2<int>*) ;

database *db;

void main(int, char **argv)
{
 db = database ::open(argv[1]);

 do_transaction() {
 void *v = NULL;
 foreach(Client *cl, client_set) {
 C_2<int> *temp = cl->number_of_persons_c2 ;
 cl->number_of_persons_demon_ptr_list = NULL_DEMON;
 cl->reservations_demon_ptr_list = NULL_DEMON;
 cl->number_of_persons_c2 = NULL;
 foreach( Reservation *r, cl->reservations(v)) {
        r->number_of_persons_demon_ptr_list = NULL_DEMON;
        r->number_of_persons_c2 = NULL;
 }
 delete temp;
 }
 }

 db->close();
}

/************ FILE const_function.cc ***********************************/
#include <ostore/ostore.hh>
#include <ostore/coll.hh>
#include <iostream.h>
#define TRUE 1
#define FALSE 0
#define NULL_DEMON 0
#define ATT_DEMON 1
#include "schema.hh"
```

```
extern void Client_number_of_persons(int _number_of_persons,
                        int new__number_of_persons,C_2<int>*);
extern void Client_reservations(Reservation *element,int mod_status);
extern void Reservation_number_of_persons(int _number_of_persons,
                        int new__number_of_persons, C_2<int>*);
extern void null_demon(char *);
extern void null_demon (Address *);
extern void null_demon (Country *) ;
extern void null_demon (Date *) ;
extern void null_demon (Tour *) ;
extern void null_demon (City *) ;
extern void null_demon (Reservation *) ;
extern void null_demon (Place_to_go *) ;
extern void null_demon (Stage *);
extern void null_demon (Hotel *) ;
extern void null_demon (Money*);
extern void null_demon (Day_price *) ;
extern void null_demon (string *) ;
extern void null_demon (Person *) ;
extern void null_demon (Client *);
extern void null_demon (float) ;
extern void null_demon (int) ;
extern void null_demon (int,int,C_2<int>*) ;
extern void null_demon (Reservation*,int,C_2<int>*) ;

extern "C" exit(int);

int
operator==(int i ,os_Set<int*> s)
{
 foreach(int *j,s)
 if(*j!=i) return(FALSE) ;
 return(TRUE) ;
}

int
operator<=(int i,os_Set<int*> s)
{
 foreach(int *j,s)
 if(*j<i) return(FALSE) ;
 return(TRUE) ;
}

int
```

```
operator>=(int i, os_Set<int*> s)
{
 foreach(int *j,s)
 if(*j>i) return(FALSE);
 return(TRUE) ;
}

int
operator<(os_Set<int*> s,int i)
{
 foreach(int *j, s)
 if(*j>=i) return(FALSE) ;
 return(TRUE) ;
}

int
operator>(int i ,os_Set<int*> s)
{
 foreach(int *j,s)
 if(*j>=i) return(FALSE) ;
 return(TRUE) ;
}

int
operator!=(int i,os_Set<int*> s)
{ foreach(int *j, s)
 if(*j==i) return(FALSE);
 return(TRUE);
}


void
SplicePath( Reservation* old_res, Reservation* new_res)
{
 C_2<int> *C1 = old_res->number_of_persons_c2;
 foreach(int *i,C1->ckval2)
 if (*i == old_res->number_of_persons()) {
        C1->ckval2.remove(i);
        break;
 };
 persistent<db> int new_value = 0;
 new_value = new_res->number_of_persons() ;
 C1->ckval2.insert(&new_value);
}
```

```
void undemonize(Reservation *r,C_2<int> *C1)
{
 r->number_of_persons_demon_ptr_list = NULL_DEMON;
 r->number_of_persons_c2 = NULL;
}

void demonize(Reservation *r,C_2<int>*C1)
{
 r->number_of_persons_demon_ptr_list = ATT_DEMON;
 r->number_of_persons_c2 = C1;
 if(r->number_of_persons()>C1->ckval1)
 printf("the inserted reservation's number of persons > client's number of persons!\n
please change it!\n");
}

void exception()
{
 printf("constraint violation!\n");
 exit(0);
}

void Hotel_reservation_end_date (Date *){};
void Employee_salary (Money *){};
void Client_returning_date (Date *){};
void Place_to_go_address (Address *){};
void Museum_artists (Person *){};
void City_places_to_go (Place_to_go *){};
void Tour_city (City *){};
void Monument_architect (Person *){};
void Place_to_go_name (string *){};
void Theater_number_of_seats (int){};
void Hotel_day_price (Day_price *){};
void Client_place_of_stay (City *){};
void Tour_price (Money *){};
void Hotel_stars (int){};
void Person_first_name (string *){};
void Hotel_reservation_type_of_stay (string *){};
void Employee_title (string *){};
void Client_departure_date (Date *){};
void Person_name_demon (string *){};
void Theater_building_date (Date *){};
void Person_birthdate (Date *){};
void City_hotels (Hotel *){};
```

```
void Country_name (string *){};
void Client_bill (float){};
void Tour_reservation_selected_tour (Tour *){};
void Country_cities (City *){};
void Place_to_go_price (Money *){};
void Person_address (Address *){};
void City_name (string *){};
void Hotel_facilities (string *){};
void Tour_description (Stage *){};
void Hotel_reservation_selected_hotel (Hotel *){};
void Reservation_date_of_reservation (Date *){};
void Museum_topic (string *){};
void Employee_seniority (int){};
void Hotel_name (string *){};
void City_country (Country *){};
void City_tours (Tour *){};
void Tour_capacity (int){};
void Tour_availability (int){};
void Monument_building_date (Date *){};
void Reservation_client (Client *){};
void Hotel_address (Address *){};
void Person_social_security (char *){};
void Tour_theme (string *){};
void Client_employee_discount (float){};
```