บทที่ 3

การออกแบบและพัฒนาโปรแกรม

การออกแบบระบบจัดการออบเจ็คต์ถาวรใช้หลักการออกแบบเชิงวัตถุและเทคนิคการเขียนโปรแกรมเชิงวัตถุของภาษาซี++ โดยการวิจัยนี้เลือกใช้โปรแกรมบอร์แลนด์ซี++ เป็นตัวแปลภาษา ระบบจัดการออบเจ็คต์ถาวรที่ได้จากการออกแบบอยู่ในลักษณะของคลาสไลบรารี การทำให้ออบเจ็คต์สามารถคงอยู่ถาวรใช้วิธีการสืบทอดจากคลาสพื้นฐาน โดยคลาสที่ต้องการคงอยู่ถาวรต้องสืบทอดมาจากคลาสพื้นฐานที่กำหนด การจัดการออบเจ็คต์ถาวรทำผ่านฟังก์ชันหรือวิธีการของคลาส

ระบบจัดการออบเจ็คต์สามารถแบ่งออกเป็น 2 ส่วนคือ ส่วนที่ใช้จัดการออบเจ็คต์และส่วนที่ติดต่อกับผู้ใช้ ซึ่งระบบที่ต้องการมีคุณลักษณะดังนี้

- สามารถเรียกออบเจ็คต์ที่มีการจัดเก็บไว้มาใช้โดยใช้ค่าคีย์ในการอ้างถึง โดยมีการตรวจสอบว่ามีออบเจ็คต์นั้นอยู่หรือไม่
- เพิ่มออบเจ็คต์ที่ยังไม่มีอยู่ โดยการกำหนดข้อมูล และบอกให้มีการจัดเก็บ
- แก้ไขออบเจ็คต์ที่มีอยู่ด้วยการอ้างถึงด้วยคีย์ เปลี่ยนค่าของข้อมูลและให้มีการจัดเก็บ
- ลบออบเจ็คต์ที่มีอยู่ออกจากหน่วยเก็บข้อมูล
- สามารถเข้าถึงออบเจ็คต์ได้โดยใช้ค่าคีย์หรือตำแหน่งที่อยู่ของออบเจ็คต์

การออกแบบโปรแกรมเกี่ยวข้องกับการออกแบบหน่วยเก็บข้อมูลและการออกแบบคลาสพื้นฐานต่างๆ ต้องรู้ว่าจะจัดการกับข้อมูลอย่างไร การนำไปใช้เกี่ยวข้องกับการกำหนดฐานข้อมูล การประกาศคลาสของออบเจ็คต์ถาวรเป็นดีไรฟ์คลาส การกำหนดสมาชิกชนิดข้อมูลชนิดฟังก์ชันและสมาชิกที่เป็นค่าคีย์ รวมทั้งการเรียกใช้ฟังก์ชันต่างๆดังมีรายละเอียดต่อไปนี้

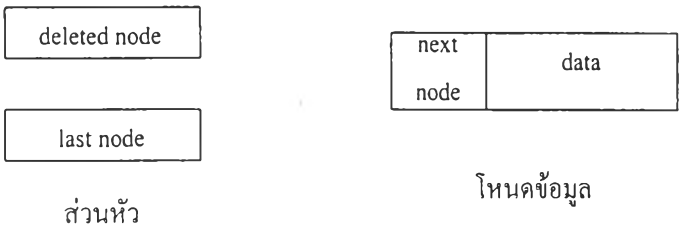
1. <u>โครงสร้างแฟ้มข้อมูล (File structure)</u>

เป็นการกำหนดโครงสร้างของแฟ้มข้อมูลและรูปแบบของระเบียนที่ระบบจัดการออบเจ็คต์ใช้จัดการกับข้อมูลบนดิสก์ โดยหน่วยเก็บข้อมูลของระบบจัดการออบเจ็คต์ประกอบด้วยแฟ้มข้อมูล 2 แฟ้มคือ แฟ้มเก็บข้อมูลของออบเจ็คต์และแฟ้มดัชนีซึ่งเก็บดัชนีชี้ไปยังออบเจ็คต์

แฟ้มข้อมูลประกอบด้วยส่วนหัวของแฟ้มอยู่ที่ต้นแฟ้มและโหนดต่างๆของข้อมูล แฟ้มข้อมูลจะถูกจัดการในลักษณะของโหนดที่มีขนาดคงที่ อ้างอิงด้วยหมายเลขโหนด ออบเจ็คต์เป็นข้อมูลที่มีขนาดไม่แน่นอนสามารถมีขนาดเกินหนึ่งโหนดได้

ระบบจะมีตัวชี้ชี้ไปยังโหนดแรก แล้วแต่ละโหนดก็จะมีตัวชี้ชี้ไปยังโหนดถัดไป เมื่อไม่มีการใช้งานโหนดใดแล้วต้องลบทิ้งโดยเอาไปไว้ในลิสต์ของโหนดที่ถูกลบ เมื่อต้องการใช้ก็จะขอมาจากระบบ โดยจะเลือกใช้โหนดที่อยู่ในลิสต์ของโหนดที่ถูกลบก่อน ถ้าไม่มีจึงใช้โหนดใหม่ต่อจากโหนดสุดท้ายที่มีอยู่

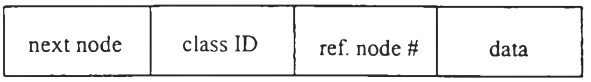
ส่วนหัวของแฟ้มประกอบด้วยเลข 2 จำนวน จำนวนแรกชี้ไปยังโหนดแรกในลิสต์ของโหนดที่ถูกลบ ส่วนจำนวนที่ 2 ชี้ไปยังโหนดสุดท้ายที่ถูกใช้งาน ในโหนดจะมีตัวชี้ไปยังโหนดถัดไปในลิสต์และมีส่วนที่ เก็บข้อมูลดังรูปที่ 3.1



รูปที่ 3.1 แสดงส่วนประกอบของแฟ้มข้อมูล

1.1 <u>ข้อมูลของออบเจ็คต์</u>

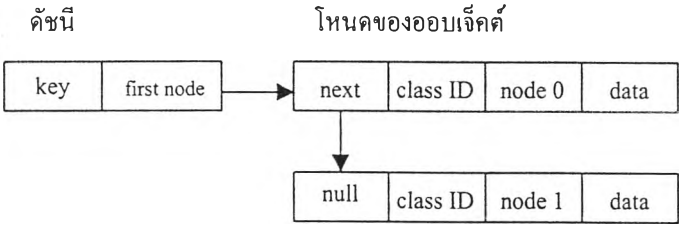
แฟ้มข้อมูลของออบเจ็คต์ประกอบด้วยส่วนหัวของแฟ้มและโหนดข้อมูลต่างๆ แต่ละออบเจ็คต์ถูกเก็บอยู่ในลิสต์ของโหนด ในโหนดจะมีส่วนหัวของออบเจ็คต์และส่วนที่เป็นข้อมูลของออบเจ็คต์ดังแสดงในรูป



รูปที่ 3.2 แสดงรูปแบบของโหนดข้อมูล

ส่วนหัวของออบเจคต์ประกอบด้วย ID ของคลาส จำนวนเลขที่บอกว่าเป็นโหนดที่เท่าไรของออบเจ็คต์นั้น โดยหมายเลขโหนดจะเป็นตำแหน่งที่อยู่ของของออบเจ็คต์ ข้อมูลของออบเจ็คต์จะอยู่ต่อมาจากส่วนหัว ถ้าออบเจ็คต์มีขนาดใหญ่กว่าหนึ่งโหนดก็จะถูกแบ่งไปเก็บในโหนดถัดไปไปจนกระทั่งเก็บได้หมด

แฟ้มข้อมูลสามารถจัดเก็บออบเจ็คต์ได้หลายชนิดและหลายขนาด แต่ต้องมีคำจำกัดความของคลาสและฟังก์ชันที่มาจัดการกับออบเจ็คต์ ระบบจัดการออบเจ็คต์จะค้นหาออบเจ็คต์ได้ต้องรู้ตำแหน่งโหนดแรกของออบเจ็คต์โดยใช้แฟ้มดัชนี สำหรับคลาสที่ไม่ต้องการใช้คีย์อาจเข้าถึงข้อมูลของออบเจ็คต์ได้โดยใช้ตำแหน่งที่อยู่หรือหมายเลขโหนดของออบเจ็คต์



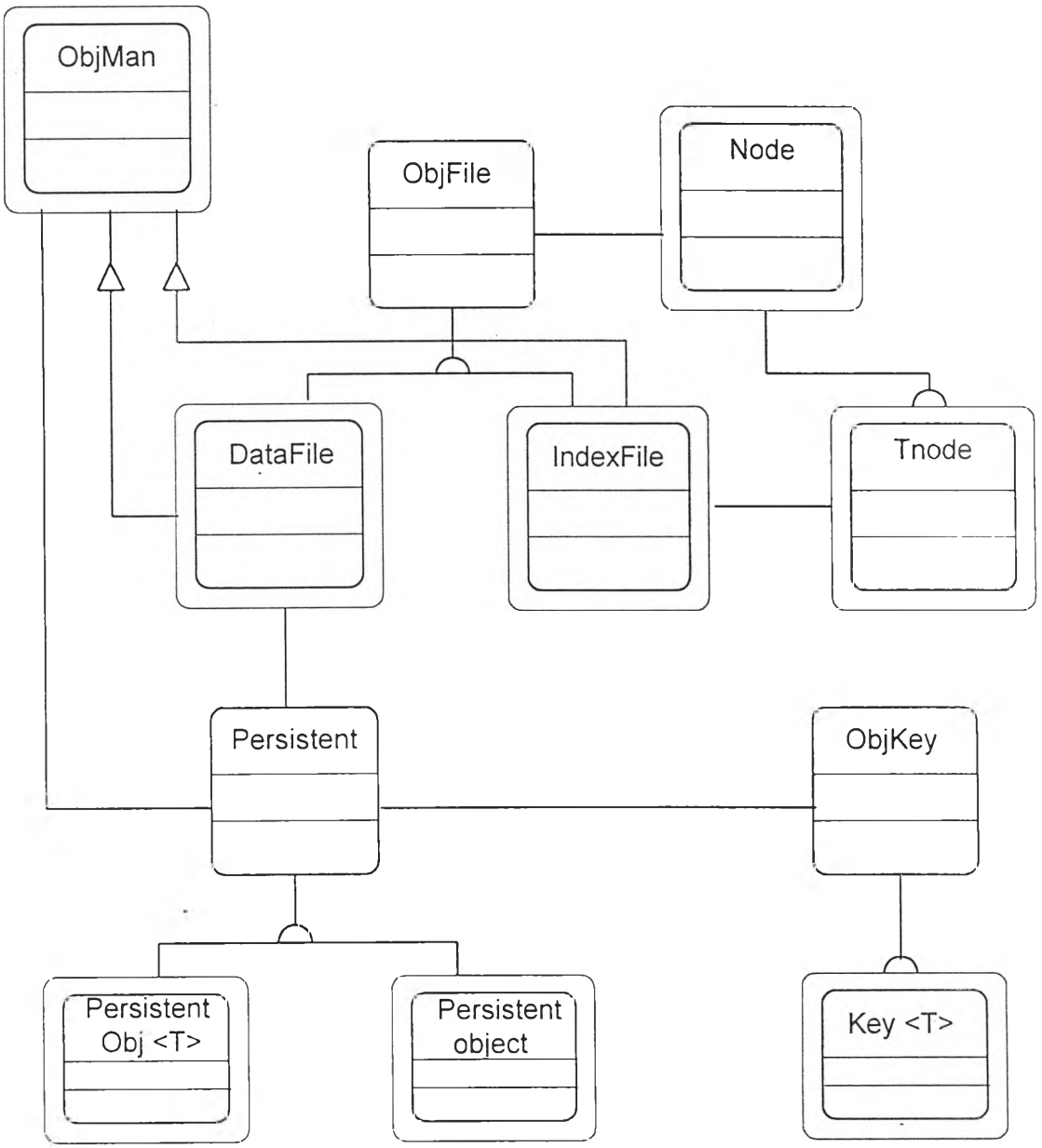
รูปที่ 3.3 แสดงความสัมพันธ์ระหว่างแฟ้มดัชนีกับแฟ้มข้อมูลของออบเจ็คต์

2. ดัชนี

แฟ้มดัชนีประกอบด้วยดัชนีที่เกี่ยวข้องกับค่าคีย์และตำแหน่งโหนดของออบเจ็คต์ในแฟ้มข้อมูล โดยจะมีบล็อคส่วนหัวสำหรับดัชนีของแต่ละคลาส บล็อคแรกในแฟ้มจะเก็บส่วนหัวของดัชนีคลาสที่มี ID เป็น 0 บล็อคที่ 2 เก็บของคลาสที่มี ID เป็น 1 ในแต่ละบล็อคเป็นอะเรย์ของส่วนหัวของดัชนีสำหรับทุกคีย์ (คีย์หลัก คีย์ที่สอง ไปเรื่อยๆ) ซึ่งเก็บหมายเลขของโหนดที่เป็นรากสำหรับดัชนี ดัชนีที่ใช้จะเป็นแบบไบนารีทรี

2. <u>โครงสร้างของคลาส</u>

ระบบจัดการออบเจ็คต์ถาวรในส่วนของการจัดการออบเจ็คต์ใช้แบบจำลองเชิงวัตถุโดยใช้สัญญลักษณ์ (notation) ของ Coad แสดงได้ดังรูปที่ 3.4



รูปที่ 3.4 แสดงแบบจำลองของระบบ

จากรูปที่ระดับล่าง PersistentObj <T> เทมเพลทใช้จัดการความคงอยู่ถาวรของคลาสที่มีโครงสร้างง่ายๆไม่มีตัวชี้และไม่ใช้คีย์ ส่วนออบเจ็คต์ถาวรซึ่งมีโครงสร้างซับซ้อนขึ้นหรือมีการใช้คีย์ต้องสืบทอดมาจากคลาส Persistent ซึ่งเป็นคลาสพื้นฐานสำหรับออบเจ็คต์ถาวรทุกคลาส ส่วน Key <T> เทมเพลทคลาสจะเพิ่มการทำงานของคีย์ดัชนี โดยมี ObjKey เป็นคลาสพื้นฐาน

ที่ระดับบนจะเป็นโครงสร้างภายในของระบบจัดการออบเจ็คต์ คลาส ObjMan จะเป็นการทำงานของระบบจัดการออบเจ็คต์ซึ่งทำหน้าที่ของฐานข้อมูล ประกอบด้วย DataFile และ IndexFile เพื่อจัดการหน่วยเก็บข้อมูลของข้อมูลและดัชนีของออบเจ็คต์ โดยคลาส DataFile และ IndexFile สืบทอดมาจากคลาส ObjFile ซึ่งจัดการอินพุทและเอาต์พุทโหนดจากดิสก์ คลาส Tnode เป็นโหนดพิเศษที่เพิ่มคุณสมบัติของดัชนีแบบบีทรี

<u>การจัดคลาสในระบบจัดการออบเจ็คต์ถาวร</u>

จากการออกแบบสามารถกำหนดคลาสต่างๆในระบบได้ดังนี้

Class	Description
class bool	เป็นข้อมูลชนิด boolean ซึ่งมีได้ 2 ค่าคือ true หรือ false
class Date	ใช้กับข้อมูลชนิดวันที่ ประกอบด้วยเดือน วัน และปี
struct Class	ใช้เก็บข้อมูลเกี่ยวกับคลาส เช่น ID ของคลาส ชื่อคลาส
struct ObjAddr	เป็นจำนวนเลขซึ่งเก็บตำแหน่งที่อยู่ของออบเจ็คต์
struct ObjHeader	เป็นส่วนหัวของแต่ละออบเจ็คต์
class Node	คลาสของโหนดข้อมูล
class Tnode	คลาสของโหนดในบีทรี สืบทอดมาจาก Node
class ObjFile	คลาสของแฟ้มข้อมูล จัดการกับอินพุทเอาต์พุทโหนดในดิสก์
class DataFile	คลาสของแฟ้มเก็บข้อมูลของออบเจ็คต์ สืบทอดมาจาก ObjFile
class IndexFile	คลาสของแฟ้มดัชนีของออบเจ็คต์ สืบทอดมาจาก ObjFile
class ObjKey	คลาสของคีย์สำหรับออบเจ็คต์ เป็นคลาสพื้นฐานให้กับเทมเพลท Key

Class	Description
template <class T> class Key	เป็นเทมเพลทคลาสสำหรับติดตั้งคีย์ให้กับคลาสของออบเจ็คต์ถาวรซึ่งดีไรฟ์จากคลาส Persistent
template <class T> class Linklist	เทมเพลทซึ่งใช้ติดตั้งโครงสร้างข้อมูลแบบลิงค์ลิสต์ 2 ทาง
template <class T> class ListEntry	เทมเพลทของโหนดต่างๆในลิงค์ลิสต์
class ObjBtree	เป็นคลาสซึ่งใช้ติดตั้งอินเด็กซ์แบบไบนารีทรี
class Objman	คลาสซึ่งทำหน้าที่ของระบบจัดการออบเจ็คต์ คอยดูแลว่ามีออบเจ็คต์ใดถูกเรียกเข้ามาในหน่วยความจำแล้ว
class Persistent	ใช้เป็นคลาสพื้นฐานให้กับคลาสของออบเจ็คต์ถาวร ดูแลลิสต์ของคีย์ของออบเจ็คต์และข้อมูลเกี่ยวกับหน่วยเก็บข้อมูลของออบเจ็คต์
Template <class T> PersistentObj	เทมเพลทซึ่งดีไรฟ์จากคลาส Persistent ใช้กับออบเจ็คต์ที่มีโครงสร้างง่ายๆ ไม่ใช้คีย์ ไม่มีการอ้างอิงและตัวชี้
class SCR	เป็นคลาสที่ใช้เชื่อมต่อระหว่างผู้ใช้ (user interface) กับโปรแกรม
class ScreenMenu	เป็นเมนูแบบทั้งจอ สามารถเลื่อนแถบสว่างหรือกดคีย์เพื่อเลือกการทำงาน
class LineMenu	เป็นเมนูแบบบรรทัดเดียว ใช้การกดคีย์อักษรตัวแรกของข้อความแสดงรายการในเมนูเพื่อเลือกการทำงาน

ตารางที่ 3.1 แสดงคลาสในระบบ

3. <u>การติดต่อกับผู้ใช้</u>

ระบบนี้การติดต่อกับผู้ใช้จะทำผ่านจอภาพและคีย์บอร์ดในลักษณะเมนู โดยต้องมีการสร้างออบเจ็คต์ของคลาส SCR ซึ่งเป็นคลาสในการติดตั้งการติดต่อกับผู้ใช้เป็นขอบเขตของจอภาพ ซึ่งในคลาสนี้มีฟังก์ชันต่างๆซึ่งเตรียมไว้ให้ใช้ได้แก่

AnyKey	รอรับการกดคีย์
ClearScr	ใช้ลบจอภาพ
Error	แสดงข้อความบอกความผิดพลาด
GetKBChar	อ่านค่าคีย์จากคีย์บอร์ด
KBWaiting	ดูว่ามีการกดคีย์หรือไม่
SetCursor	เลื่อนเคอร์เซอร์ไปยังตำแหน่งที่ต้องการ
StatusLine	แสดงข้อความที่บรรทัดส่วนท้ายของจอ
UsrInput	แสดงข้อความและอ่านข้อมูลที่ผู้ใช้ป้อนเข้ามาเก็บในตัวแปร มีโอเวอร์โหลดฟังก์ชันสำหรับข้อมูลแต่ละชนิด
WriteChar	เขียนตัวอักษรบนจอภาพยังตำแหน่งที่ระบุ
YesNo	แสดงคำถามและและรอรับการตอบ Y หรือ N จากผู้ใช้

ถ้าต้องการเมนูให้ใช้คลาส ScreenMenu ดังตัวอย่าง

```
ScreenMenu ("PERSONNEL MENU",
        "Employees " , QueryEmpl,
        "Departments " , QueryDept,
        " Projects " , QueryProj,
        NULL) .Execute( );
```

พารามิเตอร์แรกจะเป็นชื่อของเมนูและลำดับต่อมาเป็นคู่ของตัวแปรซึ่งเป็นทางเลือกของเมนูประกอบด้วยชื่อหัวข้อและฟังก์ชันที่เกี่ยวข้อง ฟังก์ชัน Execute จะทำการลบจอภาพ แสดงเมนู และคอยผู้ใช้กดคีย์ สามารถเลือกการทำงานจากเมนูโดยกดหมายเลขของหัวข้อหรือเลื่อนแถบไปยังหัวข้อที่ต้องการแล้วกดคีย์ว่าง ฟังก์ชันจะถูกเรียกมาทำงาน ใช้ ESC เพื่อออกจากการทำงานหรือไปยังเมนูก่อนหน้านั้น

ถ้าต้องการเมนูแบบบรรทัดเดียวใช้คลาส LineMenu ดังตัวอย่าง

```
LineMenu ("N-ame, D-epartment", ChangeName, ChangeDept) .Execute( );
```

พารามิเตอร์ในส่วนแรกเป็นข้อความแสดงเมนู ลำดับต่อมาเป็นฟังก์ชันที่เกี่ยวข้อง ฟังก์ชัน Execute จะแสดงข้อความของเมนูแล้วรอการกดคีย์จากผู้ใช้ สามารถเลือกการทำงานโดยคีย์อักษรตัวแรกของหัวข้อ ฟังก์ชันจะถูกเรียกมาทำงาน กด ESC เพื่อออกจากเมนู

4. การกำหนดฐานข้อมูลของระบบจัดการออบเจ็คต์

การใช้ระบบจัดการออบเจ็คต์ถาวรต้องมีการประกาศฐานข้อมูลของออบเจ็คต์ก่อน โดยประกาศออบเจ็คต์ของคลาส ObjMan ตัวอย่างการประกาศฐานข้อมูลได้แก่

```
ObjMan  * personel ;
void  Main ( )
{  personel  =  new ObjMan ( " PERSONEL" ) ;
     |
   delete  personel ;
}
```

จากตัวอย่าง PERSONEL จะเป็นชื่อของฐานข้อมูลโดยจะใช้เป็นชื่อของแฟ้มข้อมูล 2 แฟ้ม สำหรับออบเจ็คต์และคีย์คือ PERSONEL.DAT และ PERSONEL.IDX การประกาศฐานข้อมูลที่ยังไม่มีอยู่จะเป็นการสร้างฐานข้อมูลขึ้นใหม่ แต่ถ้ามีอยู่แล้วเป็นการเปิดขึ้นมาใช้งาน ฐานข้อมูลจะถูกเปิดอยู่จนกว่าจะออกนอกขอบเขตการทำงานของโปรแกรม

ระบบจัดการออบเจ็คต์ใช้คำจำกัดความของคลาสในซี++ เพื่ออธิบายสมาชิกข้อมูล สมาชิกฟังก์ชันของแต่ละคลาสและรวมเข้ากับระบบจัดการออบเจ็คต์ การออกแบบคลาสซึ่งเป็นออบเจ็คต์ถาวรต้องสืบทอดมาจากคลาสพื้นฐานในไลบรารี ซึ่งทำให้สามารถเรียกใช้ฟังก์ชันของระบบจัดการออบเจ็คต์ได้

5. การกำหนดคลาสของออบเจ็คต์ถาวร

คลาสของออบเจ็คต์ซึ่งคงอยู่ถาวรต้องสืบทอดมาจากคลาสพื้นฐาน Persistent ปกติคลาสจะมีสมาชิกข้อมูลที่เป็นค่าคีย์โดยใช้ Key<T> การออกแบบคลาสของออบเจ็คต์ถาวรต้องเตรียมคอนสตรัคเตอร์ ดีสตรัคเตอร์ของดีไรฟ์คลาสพร้อมทั้งฟังก์ชัน Read และ Write ดังตัวอย่าง

```
class  Employee  :  public  Persistent
{    Key < int >  emplno ;
     string  name ;
     void  Read ( ) ;
```

```
    void   Write ( ) ;
public :
    Employee ( int  empn ) ;
    ~ Employee ( ) ;
        |
} ;
Employee : : Employee (int  empn) :  emplno (empn)
{  . . .
   Restore ( );
}
Employee : : ~ Employee ( )
{  Store ( ) ;
    . . .
}
```

5.1 ฟังก์ชันที่เกี่ยวข้องกับการสร้างและทำลายออบเจ็คต์ถาวร

คอนสตรัคเตอร์

ลำดับการทำงานของคอนสตรัคเตอร์มีความสำคัญกับการทำงานของระบบจัดการออบเจ็คต์ถาวร การสร้างออบเจ็คต์ถาวรคอนสตรัคเตอร์ของคลาสพื้นฐานจะทำงานก่อนคอนสตรัคเตอร์ของดีไรฟ์คลาส ส่วนการทำลายออบเจ็คต์ดีสตรัคเตอร์ของคลาสพื้นฐานจะทำงานหลังจากดีไรฟ์ดีสตรัคเตอร์ทำงานเสร็จ ดังนั้นโปรแกรมจึงต้องรอจนกระทั่งออบเจ็คต์ถูกสร้างอย่างสมบูรณ์ก่อนที่จะอ่านข้อมูลขึ้นมา และต้องเขียนลงดิสก์ก่อนที่ออบเจ็คต์จะถูกทำลาย

คอนสตรัคเตอร์ของคลาสพื้นฐาน Persistent ซึ่งมีการอ้างถึงฐานข้อมูลจะทำงานเป็นอันดับแรก กำหนดค่าสมาชิกและตัวชี้เพื่อชี้ไปยังออบเจ็คต์ที่ถูกสร้าง ต่อมาเป็นคอนสตรัคเตอร์ของสมาชิกที่เป็นคีย์ของออบเจ็คต์ คอนสตรัคเตอร์จะเพิ่มคีย์เข้าไปในลิสต์ของคีย์ที่เกี่ยวข้องกับออบเจ็คต์ หลังจากนั้นจึงทำคอนสตรัคเตอร์ของดีไรฟ์คลาสเอง ซึ่งจะมีการกำหนดค่าของคีย์หลักแล้วไปเรียกฟังก์ชัน Restore ในคลาสพื้นฐานเพื่อค้นหาและโหลดออบเจ็คต์จากฐานข้อมูลเข้ามายังหน่วยความจำ

ฟังก์ชัน Restore

ฟังก์ชันจะดูว่ามีออบเจ็คต์นั้นในหน่วยความจำหรือยัง ในกรณีที่มีอยู่แล้วจะให้เอ็กเซพชัน (exception) ด้วยที่อยู่ของออบเจ็คต์นั้น ส่วนกรณีที่ไม่มีออบเจ็คต์นั้นอยู่ในหน่วยความจำก็จะไปหาออบเจ็คต์ในฐานข้อมูลด้วยค่าคีย์ ถ้ามีออบเจ็คต์อยู่ในฐานข้อมูลฟังก์ชันจะชี้ตำแหน่งในแฟ้มว่าออบเจ็คต์อยู่ที่ไหน แล้วไปเรียกฟังก์ชัน Read ของดีไรฟ์คลาสเพื่ออ่านข้อมูลของออบเจ็คต์ หากไม่มีออบเจ็คต์ในฐานข้อมูลก็จะเก็บเงื่อนไขและกลับออกมา ถ้ามีการขอจองหน่วยความจำต้องทำก่อนเรียกฟังก์ชัน Restore

ฟังก์ชัน Read และ ReadObj

ฟังก์ชัน Restore ของคลาส Persistent ไปเรียกฟังก์ชัน Read ที่กำหนดไว้ในในดีไรฟ์คลาส โดย Read เป็นเวอร์ชวลฟังก์ชันในคลาส Persistent ฟังก์ชัน Read จะอ่านสมาชิกข้อมูลเข้ามาในออบเจ็คต์ถาวรด้วยการเรียกฟังก์ชัน ReadObj จากคลาส Persistent สำหรับแต่ละข้อมูล ฟังก์ชัน ReadObj ใช้การอ้างถึงสมาชิกข้อมูล เป็นโอเวอร์โหลดฟังก์ชันในคลาส Persistent สำหรับข้อมูลแต่ละชนิด ฟังก์ชันนี้ใช้อินพุทเอาท์พุทของแฟ้มข้อมูลเพื่ออ่านตัวอักษรเข้ามายังที่อยู่ที่กำหนด สำหรับค่าคีย์ต้องอ่านข้อมูลมาไว้ในตัวแปรก่อนแล้วค่อยกำหนดให้สมาชิกที่เป็นคีย์ด้วยฟังก์ชัน SetKey ของคลาส Key ตัวอย่างแสดงฟังก์ชัน Read

```
void Employee : : Read( )
{
  int  emno;
  ReadObj(emno);
  emplno.SetKey(emno);
  ReadObj(name);
}
```

ดีสตรัคเตอร์

ส่วนการทำลายออบเจ็คต์ลำดับของดีสตรัคเตอร์มีความสำคัญ ดีสตรัคเตอร์ของดีไรฟ์คลาสจะทำเป็นอันดับแรก ซึ่งข้อมูลทั้งหมดในหน่วยความจำของออบเจ็คต์ยังอยู่ ดีสตรัคเตอร์จะไปเรียกฟังก์ชัน Store ในคลาสพื้นฐาน Persistent เพื่อจัดเก็บออบเจ็คต์ในฐานข้อมูลก่อนที่จะทำอย่างอื่น

ฟังก์ชัน Store

ถ้าต้องการเพิ่มออบเจ็คต์ที่ยังไม่มีอยู่ในฐานข้อมูล ต้องมีการเขียนออบเจ็คต์ลงฐานข้อมูล หรือเมื่อมีออบเจ็คต์อยู่แล้วและมีการเปลี่ยนแปลงข้อมูลก็ต้องเขียนออบเจ็คต์ใหม่

ฟังก์ชัน Store กำหนดตำแหน่งที่อยู่ในฐานข้อมูลให้กับออบเจ็คต์ใหม่ และเพิ่มค่าคีย์เข้าไปในดัชนีของคลาส ถ้าออบเจ็คต์มีการเปลี่ยนแปลงฟังก์ชันจะให้ตำแหน่งในแฟ้มเป็นที่อยู่ของออบเจ็คต์และแก้ไขค่าคีย์ที่เปลี่ยนแปลงในดัชนี แล้วไปเรียกฟังก์ชัน Write ของดีไรฟ์คลาสเพื่อเขียนข้อมูลของออบเจ็คต์ ถ้าต้องการลบออบเจ็คต์ที่มีอยู่ฟังก์ชัน Store จะปล่อยเนื้อที่ของออบเจ็คต์ในแฟ้ม และลบค่าคีย์ออกจากดัชนีของคลาส ถ้ามีการยกเลิกการจองหน่วยความจำต้องทำหลังเรียกฟังก์ชัน Store

ฟังก์ชัน Write และ WriteObj

ฟังก์ชัน Write เป็นเวอร์ชวลฟังก์ชันในคลาส Persistent ฟังก์ชันนี้จะไปเรียกฟังก์ชัน WriteObj ในคลาส Persistent สำหรับแต่ละสมาชิกข้อมูลในออบเจ็คต์ โดยลำดับการเรียกต้องเหมือนกับการเรียก ReadObj ในฟังก์ชัน Read

WriteObj เป็นโอเวอร์โหลดฟังก์ชันสำหรับแต่ละชนิดข้อมูล ฟังก์ชันใช้อินพุทเอาท์พุทของแฟ้มข้อมูลเพื่อเขียนตัวอักษรจากตำแหน่งที่อยู่ที่ระบุ โดยเขียนตามโหนดของออบเจ็คต์และมีการสร้างโหนดใหม่ถ้าจำเป็น ใช้ KeyValue ซึ่งเป็นฟังก์ชันของคลาส Key เพื่อใช้แสดงค่าคีย์ ตัวอย่างแสดงฟังก์ชัน Write

```
void Employee : : Write( )
{
  WriteObj(emplno.KeyValue( ));
  WriteObj(name);
}
```

ส่วนการทำลายคีย์ดีสตรัคเตอร์ของคีย์จะทำหลังจากดีสตรัคเตอร์ของดีไรฟ์คลาส หลังจากนั้นดีสตรัคเตอร์ของคลาส Persistent จะทำงานหลังสุด ซึ่งทำให้ออบเจ็คต์ถูกโปรแกรมทำลายพร้อมที่จะเรียกคืนหรือลบจากฐานข้อมูล

5.2 การกำหนดคีย์

ควรกำหนดคีย์ให้เป็นสมาชิกแรกของออบเจ็คต์ ด้วยการใช้เทมเพลท Key เพื่อให้มีคุณสมบัติของคีย์ ถือว่าคีย์จะเป็นสมาชิกหนึ่งของคลาส ค่าของคีย์เป็นข้อมูลที่มีความยาวจำกัดซึ่งจำเป็นต่อการทำงานแบบบีทรี ในคลาสหนึ่งสามารถมีได้หลายคีย์ ตำแหน่งของคีย์ในออบเจ็คต์มีความสำคัญ คีย์แรกที่ถูกประกาศในคลาสจะเป็นคีย์หลัก ส่วนคีย์อื่นๆจะเป็นคีย์รองหมด ระบบจัดการออบเจ็คต์จะจัดการดูแลคีย์หลักและคีย์รองต่างกัน

สำหรับคีย์ที่เป็นข้อความ เมื่อคีย์เป็นออบเจ็คต์ของคลาสสตริง ระบบจัดการออบเจ็คต์มีฟังก์ชัน Key<string> ไว้ให้ การใช้งานต้องกำหนดความยาวคงที่ของคีย์ในคอนสตรัคเตอร์ โดยใช้ฟังก์ชัน SetKeylength ของ Key<string> ก่อนเรียกฟังก์ชัน Restore ดังตัวอย่าง

```
class Employee : public Persistent {
  Key<string> name;
  void Read( );
  void Write( );
public :
  Employee(const string &nm) : name(nm)
    { name.SetKeylength(20); Restore( ) ; }
};
```

5.3 การจัดเก็บออบเจ็คต์ถาวรที่มีโครงสร้างง่ายๆ

ใช้เทมเพลท PersistentObj<T> เป็นคลาสพื้นฐานให้กับออบเจ็คต์ถาวรที่มีโครงสร้างง่ายๆ ซึ่งไม่มีคีย์ ไม่มีตัวชี้ การอ้างอิง หรือเวอร์ชวลฟังก์ชัน โดยใช้ค่าตำแหน่งที่อยู่ ObjAddr สำหรับการอ้างอิง ตัวอย่างการใช้ได้แก่

```
struct Address {
  char street[20];
  char city[20];
  int zipcode; }
main( )
{ Address add1 = {“Orchard”, “Singapore”, 105};
```

```
  PersistentObj<Address> addr(add1);
  addr.Obj.zipcode = 110;
  ObjAddr oa = addr.AddObj( );    // add object to database
  PersistentObj<Address> oad(oa)  // retrieve in other time
  return 0;
}
```

6. การใช้งานระบบจัดการออบเจ็คต์ถาวร

เมื่อเปิดฐานข้อมูลแล้วสามารถเพิ่มออบเจ็คต์ถาวร เรียกออบเจ็คต์มาใช้ เปลี่ยนแปลงแก้ไข และลบออบเจ็คต์ได้ สามารถท่องไประหว่างออบเจ็คต์ของคลาสได้ ตัวอย่างคลาสของออบเจ็คต์ถาวรได้แก่

```
typedef int EmplNumber;
typedef int DeptNumber;
class Employee : pulic Persistent
{
  Key<EmplNumber> emplno;
  Key<DeptNumber> deptno;
public :
  Employee(EmplNumber en = 0);
  ~Employee( );
  Key<EmplNumber> * EmplKey( )  { return &emplno; }
  Key<DeptNumber> * DeptKey( )  { return &deptno; }
  SetEmplNo(EmplNumber en)  { emplno.KeyValue( ) = en; }
  EmplNumber EmplNo( ) const  { return emplno.KeyValue( ); }
  SetDeptNo(DeptNumber dn)  { deptno.KeyValue( ) = dn; }
  DeptNumber DeptNo( ) const  { return deptno.KeyValue( ); }
};
```

การใช้งานออบเจ็คต์ถาวรเกี่ยวข้องกับการจัดการต่างๆดังต่อไปนี้

6.1 การทำงานกับคีย์

จากคลาสตัวอย่างมี 2 คีย์ คีย์แรกเป็นคีย์หลักของคลาสอีกคีย์เป็นคีย์รอง โดยคีย์เป็นชนิดของ Key<T> เทมเพลทคลาสและเป็นสมาชิกข้อมูลส่วนตัว การใช้คีย์จำเป็นต้องกำหนดค่าให้คีย์และอ่านค่ากลับมาให้ได้ คอนสตรัคเตอร์ของคลาสกำหนดค่าเริ่มต้นให้คีย์ emplno และคีย์ deptno โดย

```
Employee : : Employee(EmplNumber en) : emplno(en), deptno(0)
{  // . . .
}
```

ต่อมาอาจมีการกำหนดค่าให้คีย์ด้วยฟังก์ชัน SetKey

```
deptno.SetKey(100);
```

ถ้าต้องการได้ค่าคีย์ให้ใช้ฟังก์ชัน KeyValue

```
DeptNumber dn = deptno.KeyValue( );
```

6.2 การเรียกออบเจ็คต์ถาวรมาใช้

สามารถเรียกออบเจ็คต์ถาวรได้โดยประกาศออบเจ็คต์ที่มีอยู่ในฐานข้อมูลดังตัวอย่าง

```
Employee empl(123);
if (empl.ObjectExist( )) {
  // object exists . . .
}
```

สร้างออบเจ็คต์ empl จากคีย์หลักเลขประจำตัวพนักงาน 123 ฟังก์ชัน ObjectExist จะให้ค่าถูกถ้าพบออบเจ็คต์ในฐานข้อมูล ออบเจ็คต์ถูกทำให้เกิดขึ้น และข้อมูลของออบเจ็คต์จะถูกอ่านเข้ามาหน่วยความจำโดยดีไรฟ์ฟังก์ชัน Read ซึ่งเตรียมไว้ ถ้าไม่มีออบเจ็คต์ในฐานข้อมูลออบเจ็คต์จะถูกสร้างขึ้นด้วยค่าคีย์ที่ระบุ ส่วนข้อมูลอื่นถูกกำหนดค่าเริ่มต้นด้วยคอนสตรัคเตอร์

หลังจากออบเจ็คต์ถูกทำให้เกิดขึ้น อาจมีการเปลี่ยนแปลงข้อมูลของออบเจ็คต์แล้วเรียกฟังก์ชันเพื่อเพิ่ม เปลี่ยนแปลงแก้ไข หรือลบออบเจ็คต์ หรือถ้าไม่มีการเรียกฟังก์ชันเหล่านี้ทำให้ออบเจ็คต์ออกนอกขอบเขตโดยไม่มีการเก็บในฐานข้อมูล

6.3 การสร้างออบเจ็คต์ถาวร

การสร้างออบเจ็คต์ถาวรทำได้โดยประกาศออบเจ็คต์ที่ยังไม่มีอยู่ในฐานข้อมูล และบอกระบบจัดการออบเจ็คต์ให้เพิ่มเข้าไปในฐานข้อมูลดังตัวอย่าง

```
Employee empl(123);
if (!empl.ObjectExist( )) {
  // object doesn't exist . . .
  empl.AddObj( );
}
```

ออบเจ็คต์ empl ถูกสร้างจากค่าคีย์หลัก 123 เมื่อระบบจัดการออบเจ็คต์หาออบเจ็คต์ที่ตรงกันไม่เจอจึงเรียกฟังก์ชัน AddObj ให้เพิ่มออบเจ็คต์ในฐานข้อมูล ออบเจ็คต์จะถูกเพิ่มในฐานข้อมูลเมื่อออกนอกขอบเขต สามารถแก้ไขออบเจ็คต์ได้อีกก่อนมีการเขียนลงฐานข้อมูล

6.4 การเปลี่ยนแปลงแก้ไขออบเจ็คต์ถาวร

สามารถเปลี่ยนแปลงแก้ไขออบเจ็คต์ถาวรได้โดยประกาศออบเจคต์ที่มีอยู่แล้ว มีการเปลี่ยนค่าข้อมูลและเรียกฟังก์ชัน ChangeObj ดังตัวอย่าง

```
Employee empl(123);
if (empl.ObjectExist( )) {
  // object exists. . . change data
  empl.ChangeObj( );
}
```

การเรียกฟังก์ชันนี้เป็นการบอกว่าออบเจ็คต์มีการเปลี่ยนแปลง และให้เขียนออบเจ็คต์ลงฐานข้อมูล การเปลี่ยนแปลงจะถูกเขียนเมื่อออกนอกขอบเขต

6.5 การลบออบเจ็คต์ถาวร

สามารถลบออบเจ็คต์ถาวรได้โดยประกาศออบเจ็คต์ที่มีอยู่แล้ว และเรียกฟังก์ชัน DeleteObj ดังตัวอย่าง

```
Employee empl(123);
if (empl.ObjectExist( )) {
  // object exists . . .
  empl.DeleteObj( ); }
```

การเรียกฟังก์ชันนี้เป็นการบอกว่าต้องการลบออบเจ็คต์จากฐานข้อมูลเมื่อออกนอกขอบเขต ยังสามารถทำงานกับออบเจ็คต์ในหน่วยความจำจนกว่าออบเจ็คต์จะออกนอกขอบเขต

7. การท่องไปในฐานข้อมูล

สามารถท่องไปในฐานข้อมูลได้ 2 วิธีคือ การเข้าถึงโดยใช้ค่าคีย์ดัชนีไปยังที่อยู่ของออบเจ็คต์เป็นแบบเชิงสัมพันธ์ หรือการเข้าถึงโดยใช้ตำแหน่งที่อยู่ของออบเจ็คต์เป็นแบบเชิงวัตถุ

7.1 การท่องไปด้วยคีย์

ใช้คีย์ดัชนีเพื่อหาออบเจ็คต์ โดยสามารถหาออบเจ็คต์ด้วยค่าคีย์หลักหรือคีย์รองได้ ฟังก์ชันที่ใช้ได้แก่

FindObj

ฟังก์ชันนี้ใช้ค้นหาออบเจ็คต์ ในตัวอย่างใช้ค่าที่กำหนดให้คีย์หลักเพื่อหาพนักงาน

```
Employee empl;
empl.SetEmplNo(123);  // initialize emplno
empl.FindObj(empl.EmplKey( ));
if (empl.ObjectExist( )) {
//  . . .
}
```

ออบเจ็คต์ Employee ถูกทำให้เกิดขึ้นโดยไม่ระบุคีย์หลัก คอนสตรัคเตอร์จะสร้างออบเจ็คต์ที่มีคีย์ว่างโดยไม่ต้องหาออบเจ็คต์ในฐานข้อมูล แล้วกำหนดค่าข้อมูลรหัสพนักงานซึ่งเป็นคีย์หลักเพื่อหาออบเจ็คต์ถาวร

ฟังก์ชัน FindObj ได้รับตัวชี้ไปยังคลาส Key เพื่อดึงข้อมูล ถ้ามีออบเจ็คต์ Employee ในฐานข้อมูลที่มีข้อมูลตรงกับค่าคีย์รหัสพนักงานนี้ ระบบจัดการออบเจ็คต์จะกำหนดค่าออบเจ็คต์ empl ด้วยออบเจ็คต์ Employee ที่เจอ

FirstObj และ LastObj

สามารถดึงออบเจ็คต์แรกหรือออบเจ็คต์สุดท้ายในลำดับของคีย์โดยใช้ฟังก์ชัน FirstObj และ LastObj ซึ่งฟังก์ชันเหล่านี้ต้องกำหนดค่าเริ่มต้นของคีย์ โดยจะดึงออบเจ็คต์จากลำดับเริ่มต้นหรือสุดท้ายของคีย์ ตัวอย่างแสดงการใช้ฟังก์ชัน

```
Employee empl;
empl.FirstObj(empl.DeptKey( ));  หรือ empl.LastObj(empl.DeptKey( ));
if (empl.ObjectExist( )) {
```

```
// object exists . . .
}
```

ฟังก์ชันส่งค่าผิดถ้าไม่มีออบเจ็คต์ของคลาสในฐานข้อมูลซึ่งมีค่าคีย์รองไม่เป็นค่าว่าง ถ้าเรียกฟังก์ชันโดยไม่ผ่านค่าตัวชี้ไปยังคีย์จะเป็นการหาออบเจ็คต์แรกหรือสุดท้ายในลำดับของคีย์

NextObj และ PrevObj

ฟังก์ชันจะส่งกลับออบเจ็คต์ถัดไปหรือก่อนหน้าในลำดับของคีย์ซึ่งถูกเรียก ฟังก์ชันเหล่านี้ต้องการตำแหน่งของคีย์ในออบเจ็คต์ ไม่เช่นนั้นจะทำงานเหมือน FirstObj และ LastObj

การท่องไปใช้ออบเจ็คต์เดียวกันในหน่วยความจำเพื่อเก็บค่าออบเจ็คต์ที่เรียกมาได้ ถ้ายังมีออบเจ็คต์ถัดไปหรือก่อนหน้าในลำดับของคีย์ ObjectExist จะให้ค่าถูก และจะให้ค่าผิดเมื่อยังมีการเรียก NextObj อีกหลังจากมาถึงออบเจ็คต์สุดท้ายแล้ว หรืออยู่ที่ออบเจ็คต์แรกแล้วยังมีการเรียก PrevObj อีก โดยแต่ละคีย์ในออบเจ็คต์ถาวรเก็บตำแหน่งของออบเจ็คต์ตามลำดับ

```
Employee empl;
empl.FirstObj(empl.DeptKey( )); หรือ empl.LastObj(empl.DeptKey( ));
while (empl.ObjectExist( )) {
  // object exists . . . process object
empl.NextObj(empl.DeptKey( )); หรือ empl.PrevObj(empl.DeptKey( ));
}
```

7.2 <u>การท่องไประหว่างคลาส</u>

บางครั้งโปรแกรมต้องเรียกออบเจ็คต์ถาวรขึ้นมาเมื่อเกี่ยวข้องออบเจ็คต์ของคลาสอื่น จากตัวอย่างต้องการพนักงานที่สังกัดกับแผนก โปรแกรมทำให้เกิดออบเจ็คต์ Department ที่มีค่าคีย์หลักเป็น 123 แล้วสร้างออบเจ็คต์ Employee ว่างๆ โดยตั้งรหัสแผนกเดียวกันให้กับคีย์รองของออบเจ็คต์และเรียก FindObj ด้วยการใช้คีย์รอง สามารถทำงานกับออบเจ็คต์ Employee นั้นและออบเจ็คต์ถัดมาด้วย NextObj จนกว่ารหัสแผนกในออบเจ็คต์ถัดไปเปลี่ยนค่า หรือ ObjectExist ให้ค่าผิดเป็นการบอกว่าสิ้นสุดลำดับของคีย์รหัสแผนกของคลาส Employee

```
Department dept(123);
if (dept.ObjectExits( )) {
  Employee empl;
  empl.SetDeptNo(123);
```

```
    empl.FindObj(empl.DeptKey( ));
    while (empl.ObjectExist( ) && empl.DeptNo( ) == deptNo( )) {
      // process . . .
      empl.NextObj(empl.DeptKey( )); }
  }
```

7.3 การท่องไปด้วยตำแหน่งที่อยู่

บางครั้งการเข้าถึงข้อมูลด้วยคีย์ดัชนีไม่ใช่วิธีที่ดีที่สุด อาจต้องการดึงออบเจ็คต์ถาวรด้วยค่าซึ่งระบบจัดการออบเจ็คต์กำหนดให้ตอนสร้างออบเจ็คต์ บางครั้งต้องการคงออบเจ็คต์ระหว่างการเปลี่ยนแปลงของโปรแกรม หรือผ่านออบเจ็คต์จากโปรแกรมหนึ่งไปยังอีกโปรแกรม ตัวอย่างเช่น มีออบเจ็คต์จำนวนไม่มากในแต่ละคลาส ซึ่งโปรแกรมต้องการเพียงหนึ่งออบเจ็คต์ของคลาส ก็ไม่จำเป็นต้องดึงออบเจ็คต์จากค่าข้อมูล การออกแบบควรรู้ว่าโปรแกรมประยุกต์ควรใช้แบบของข้อมูลแบบใด หลักการทั่วไปถ้าฐานข้อมูลเก็บออบเจ็คต์ของคลาสเดียวกันจำนวนเยอะๆควรดัชนีด้วยคีย์ ถ้าเก็บออบเจ็คต์ในคลาสเดียวกันจำนวนน้อย โปรแกรมรู้ตำแหน่งที่อยู่ด้วยค่าข้อมูลออบเจ็คต์ก็อาจไม่ต้องมีคีย์

8. การออกแบบคลาสที่ไม่ใช้คีย์

ระบบจัดการออบเจ็คต์สนับสนุนการออกแบบฐานข้อมูลและดึงออบเจ็คต์โดยไม่ต้องใช้ดัชนี ถ้าไม่มีสมาชิกข้อมูลที่สร้างด้วย Key<T> ออบเจ็คต์จะไม่มีคีย์ ซึ่งต้องระบุถึงออบเจ็คต์ด้วยตำแหน่งที่อยู่ โดยในคลาสยังคงต้องมีฟังก์ชัน Read และ Write คอนสตรัคเตอร์ก็ต้องเรียกฟังก์ชัน Restore และดีสตรัคเตอร์เรียกฟังก์ชัน Store

คลาสที่ไม่มีคีย์ต้องมีคอนสตรัคเตอร์ที่ใช้พารามิเตอร์แสดงตำแหน่งที่อยู่ของออบเจ็คต์ ObjAddr ซึ่งยอมให้ใช้โดยไม่กำหนดค่าพารามิเตอร์ได้โดยจะถือว่าค่าเป็น 0 คอนสตรัคเตอร์เรียก Restore โดยมี ObjAddr เป็นพารามิเตอร์ ตัวอย่างคลาสที่ไม่มีคีย์เช่น

```
class Text : public Persistent
{ string text;
  void Read( ) { ReadObj(text)); }
  void Write( ) { WriteObj(text)); }
```

```
public :
  Text(const string& str) : text(str) { Restore( ); }
  Text( Restore(objaddr); }
  ~Text( ) { Store( ); }
  const string& GetText( ) { Return text; }
  // . . .
};
```

8.1 <u>การสร้างออบเจ็คต์ที่ไม่มีคีย์</u>

สร้างออบเจ็คต์ไม่มีคีย์โดยทำให้เกิดขึ้นแล้วกำหนดค่าให้สมาชิกข้อมูล และเรียกฟังกชัน AddObj เช่นเดียวกับออบเจ็คต์ที่มีคีย์ ซึ่งจะได้ตำแหน่งที่อยู่ของออบเจ็คต์ใหม่นี้ ฟังก์ชัน ObjAddress ให้ค่าที่อยู่ของออบเจ็คต์ที่ถูกเพิ่มเข้าไปในฐานข้อมูล โดยโปรแกรมต้องจำที่อยู่นี้ไว้เพื่อใช้ในการดึงออบเจ็คต์ในเวลาต่อมา ตัวอย่างแสดงการสร้างออบเจ็คต์ของคลาส Text

```
Text tx("Birthday");
Tx.AddObj( );
ObjAddr oa = tx.ObjectAddress( );
```

8.2 <u>การเรียกด้วยตำแหน่งที่อยู่ของออบเจ็คต์</u>

การดึงออบเจ็คต์ที่ไม่มีคีย์จะใช้ที่อยู่เป็นพารามิเตอร์ ตัวอย่างแสดงการดึงออบเจ็คต์ Text จากฐานข้อมูล

```
void Fd(ObjAddr objadr)
{ Text tx(objadr);
  cout << tx.GetText( );
}
```

ต้องผ่านค่าที่อยู่ที่ถูกต้องให้ฟังก์ชัน ถ้าที่อยู่ไม่ได้ชี้ไปยังออบเจ็คต์ของคลาสนั้น ก็จะไม่เรียกฟังก์ชัน Read หรือ ObjectExist จะให้ค่าผิด

8.3 การท่องไปด้วยตำแหน่งที่อยู่ของออบเจ็คต์

ฟังก์ชันจะให้การอ้างอิงไปยังออบเจ็คต์ที่ต้องการ สามารถใช้ฟังก์ชัน FirstObj, LastObj, NextObj และ PrevObj กับคลาสที่ไม่มีคีย์ โดยฟังก์ชันเหล่านี้จะทำตามลำดับออบเจ็คต์ของคลาสในฐานข้อมูล ซึ่งอาจไม่ใช่ตามลำดับของออบเจ็คต์ที่ถูกสร้าง เพราะออบเจ็คต์ใหม่จะใช้เนื้อที่ของออบเจ็คต์ที่ถูกลบก่อน

การท่องไปแบบนี้ในฐานข้อมูลขนาดใหญ่จะไม่มีประสิทธิภาพ ระบบจัดการออบเจ็คต์ไม่เก็บตัวชี้ไปหรือการเชื่อมโยงของออบเจ็คต์ที่ไม่มีคีย์ เช่น เมื่อเรียก FirstObj ก็จะดูในฐานข้อมูลตั้งแต่ส่วนต้นเพื่อหาโหนดแรกของออบเจ็คต์แรกของคลาส

8.4 การแก้ไขเปลี่ยนแปลงและแก้ไขออบเจ็คต์

การแก้ไขและเปลี่ยนแปลงออบเจ็คต์ที่ไม่มีคีย์เกี่ยวข้องกับการดึงออบเจ็คต์เพื่อเปลี่ยนแปลงหรือลบ แล้วเปลี่ยนค่าสมาชิกข้อมูลและเรียก ChangeObj หรือ DeleteObj เช่นเดียวกับออบเจ็คต์ที่มีคีย์ แต่ต้องแน่ใจว่าที่อยู่ชี้ไปยังออบเจ็คต์ของคลาสที่ต้องการลบหรือแก้ไข