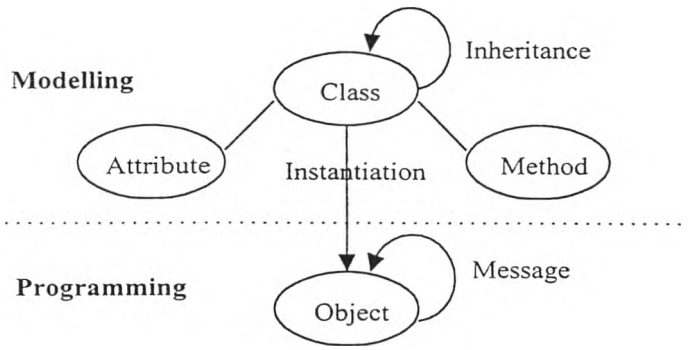
# บทที่ 2

# ทฤษฎีที่เกี่ยวข้อง

ในการออกแบบและพัฒนาระบบจัดการออบเจ็คต์ถาวรต้องใช้ทฤษฎีพื้นฐานเกี่ยวกับออบเจ็คต์ ภาษาโปรแกรมเชิงวัตถุ วิธีการการจัดเก็บออบเจ็คต์ ระบบจัดการฐานข้อมูลของออบเจ็คต์ วิธีการเข้าถึงข้อมูล และการออกแบบเชิงวัตถุ ซึ่งจะแสดงรายละเอียดดังต่อไปนี้

## 1. ภาษาโปรแกรมเชิงวัตถุ

การพัฒนาโปรแกรมด้วยภาษาโปรแกรมเชิงวัตถุสามารถแบ่งเป็นส่วนการทำงานได้ 2 ส่วน คือ การกำหนดโมเดล (Modelling) และการเขียนโปรแกรม (Programming) ดังแสดงในรูปที่ 2.1




รูปที่ 2.1 แสดงโมเดลของภาษาโปรแกรมเชิงวัตถุ

โมเดลของออบเจ็คต์รวมข้อมูลและวิธีการทำงานเข้าด้วยกันในคลาส การพัฒนาโปรแกรมเชิงวัตถุแบ่งออกเป็น

- การกำหนดคลาส ใช้การสืบทอดแอททริบิวต์และวิธีการ ส่วนนี้เกี่ยวข้องกับโมเดลของโปรแกรม
- การติดตั้งใช้งานออบเจ็คต์ โดยใช้การทำให้เกิดขึ้น (instantiation) และการส่งข่าวสารเพื่อระบุการทำงานต่างๆในระบบ ส่วนนี้เกี่ยวข้องกับการเขียนโปรแกรม

คลาสเป็นการกำหนดกลุ่มของออบเจ็คต์ที่มีคุณสมบัติคล้ายคลึงกันเข้าด้วยกัน คุณสมบัติเหล่านี้อยู่ในรูปของแอททริบิวต์และวิธีการ ออบเจ็คต์หนึ่งมีความแตกต่างจากออบเจ็คต์อื่นด้วยชื่อของออบเจ็คต์ ค่าของแอททริบิวต์ รวมทั้งวิธีการทำงาน

ออบเจ็คต์ถูกสร้างขึ้นด้วย instantiation ตามโมเดลของคลาส โดยมีการทำงานที่เป็นอิสระ คุณสมบัติและพฤติกรรมของออบเจ็คต์กำหนดด้วยคลาส แต่ละออบเจ็คต์มีการตอบสนองเมื่อได้รับข่าวสารและมีการส่งข่าวสารไปยังออบเจ็คต์อื่น ส่วนสำคัญของการเขียนโปรแกรมเชิงวัตถุคือการทำให้ออบเจ็คต์เกิดขึ้นและการส่งข่าวสารไปยังออบเจ็คต์เหล่านี้

ข่าวสารที่ส่งระหว่างออบเจ็คต์เกี่ยวข้องกับวิธีการในระดับคลาสและค่าพารามิเตอร์ที่ส่งผ่าน เมื่อออบเจ็คต์ได้รับข่าวสารจะใช้คุณลักษณะของออบเจ็คต์เพื่อทำกระบวนการตามข่าวสาร ออบเจ็คต์ถือเป็นตัวตนของคลาส มีพื้นฐานบนการส่งข่าวสาร แต่การตอบสนองจะขึ้นอยู่กับข้อกำหนดของคลาส คุณสมบัตินี้ในภาษาโปรแกรมเชิงวัตถุเรียกว่าโพลิมอร์ฟิซึม เมื่อออบเจ็คต์ได้รับข่าวสาร วิธีการที่ตอบสนองจะเปลี่ยนแปลงตามชนิดของออบเจ็คต์ สำหรับคลาสที่เกิดจากการสืบทอดถ้าคลาสของออบเจ็คต์เองไม่มีวิธีการในการตอบสนองวิธีการในคลาสพื้นฐานจะถูกใช้งาน

ลักษณะสำคัญอย่างหนึ่งของการเขียนโปรแกรมเชิงวัตถุก็คือการสืบทอดคุณสมบัติ ซี++ ใช้การสืบทอดนี้ด้วยการดีไรฟ์คลาส (class derivation) ซึ่งทำให้มีการกำเนิดคลาสใหม่จากคลาสที่มีอยู่แล้ว ประโยชน์ของการสืบทอดคุณสมบัติคือ การนำโปรแกรมที่มีอยู่แล้วมาใช้ การปรับปรุงโปรแกรมที่ทำงานคล้ายกันแต่มีความแตกต่างกันเล็กน้อย เป็นการแยกส่วนที่เป็นพื้นฐานของคลาสต่างๆออกมา และการจัดโครงสร้างของออบเจ็คต์เป็นลำดับชั้น

ซี++ มีทั้งการสืบทอดจากคลาสพื้นฐานเพียงคลาสเดียว (single inheritance) และการสืบทอดจากคลาสพื้นฐานหลายคลาส (multiple inheritance) โดยกำหนดคลาสใหม่ให้มีการสืบทอดจากคลาสที่มีอยู่แล้ว คลาสใหม่เรียกว่าดีไรฟ์คลาสจะมีสมาชิกจากคลาสพื้นฐานและสมาชิกของคลาสนั้นเอง ขอบเขตของดีไรฟ์คลาสอยู่ภายในขอบเขตของคลาสพื้นฐาน การค้นหาจะทำจากขอบเขตภายในออกไป การสร้างออบเจ็คต์คอนสตรัคเตอร์ของคลาสพื้นฐานจะถูกเรียกก่อนคอนสตรัคเตอร์ของดีไรฟ์คลาสเอง ส่วนการทำลายออบเจ็คต์ดีสตรัคเตอร์ของดีไรฟ์คลาสจะถูกเรียกก่อนดีสตรัคเตอร์ของคลาสพื้นฐาน

การเข้าถึงข้อมูลส่วนของดีไรฟ์คลาสเองเป็นไปตามปกติ แต่การเข้าถึงข้อมูลส่วนที่สืบทอดมาจากคลาสพื้นฐานโดยสมาชิกของดีไรฟ์คลาสและเฟรนด์ ไม่สามารถเข้าถึงสมาชิกส่วนตัวในคลาสพื้นฐาน ส่วนการเข้าถึงสมาชิกส่วนที่สืบทอดของคลาสพื้นฐานจากภายนอกขึ้นอยู่กับชนิดของการสืบทอดนั้น ถ้าสืบทอดแบบสาธารณะสมาชิกที่สืบทอดมาจากคลาสพื้นฐานที่เป็นสาธารณะทั้งหมด สามารถเข้าถึงได้เหมือนกับเป็นสมาชิกสาธารณะของดีไรฟ์

คลาส การสืบทอดแบบป้องกันสมาชิกที่สืบทอดมาชนิดสาธารณะและป้องกันสามารถเข้าถึงเหมือนกับเป็นสมาชิกส่วนป้องกันของดีไรฟ์คลาส ถ้าสืบทอดแบบส่วนตัวจะไม่สามารถเข้าถึงสมาชิกได้เลย

**2. ตัวชี้ (Pointer) ในซี++**

ตัวแปลภาษาของซี++ อาจมีการเพิ่มตัวชี้เข้าไปในออบเจ็คต์ ออบเจ็คต์ของซี++ สามารถมีตัวชี้ได้ 3 ชนิดคือ

1. ตัวชี้ธรรมดาใช้แสดงความสัมพันธ์ เช่น ตัวชี้แสดงความสัมพันธ์ระหว่างออบเจ็คต์การทำลิงค์ลิสต์ และทรี เป็นต้น ตัวชี้เหล่านี้อยู่ภายใต้การควบคุมของโปรแกรม

2. ตัวชี้ของเวอร์ชวลฟังก์ชัน จะเหมือนกันสำหรับทุกออบเจ็คต์ในคลาสเดียวกันในโปรแกรมเดียวกัน ตัวชี้ชนิดนี้สนับสนุนโพลิมอร์ฟิซึมของซี++ โดยตัวชี้จะถูกซ่อนเอาไว้และจัดการโดยตัวแปลภาษา

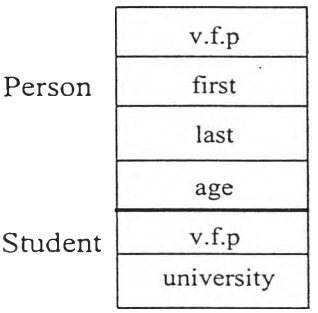
3. ตัวชี้ของเวอร์ชวลคลาส ซึ่งใช้ชี้ภายในออบเจ็คต์เดียวกัน ตัวชี้เหล่านี้ก็ถูกซ่อนไว้และจัดการด้วยตัวแปลภาษา

ขนาดของตัวชี้และตำแหน่งที่อยู่ของตัวชี้แบบเวอร์ชวลในออบเจ็คต์ขึ้นอยู่กับตัวแปลภาษาแต่ละตัว

ตัวอย่างการใช้เวอร์ชวลฟังก์ชัน

```
class person  {
public :
    char  first[MAX] ;
    char  last[MAX] ;
    int  age ;
    virtual void print ( ) ;
} ;
class student :  public person {
public :
    char  university[MAX] ;
    virtual void print ( ) ;
} ;
void person : : print ( )
```

```
{
   cout << first << " " << last << " age = " << age << endl ;
}
void student : : print ( )
{
   person : : print ( ) ;
   cout << "student at " << university << endl ;
}
```




**Student**

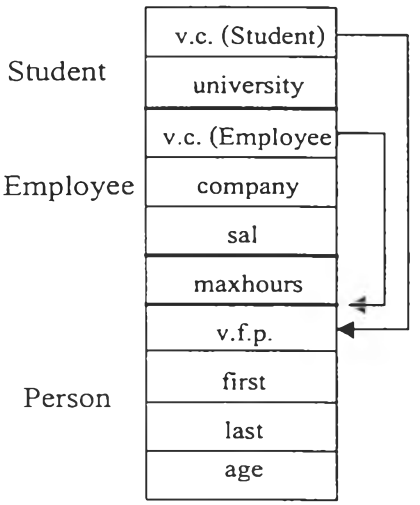
รูปที่ 2.2 แสดงเวอร์ชวลฟังก์ชัน

ตัวชี้ของเวอร์ชวลฟังก์ชันใช้ชี้ไปที่ตารางเวอร์ชวลฟังก์ชันซึ่งเก็บตัวชี้ไปยังฟังก์ชันต่างๆ การใช้งานเวอร์ชวลฟังก์ชันจะแตกต่างกันขึ้นอยู่กับชนิดของออบเจ็คต์ที่เรียกใช้ฟังก์ชัน จากตัวอย่างคลาส Person และคลาส Student มีเวอร์ชวลฟังก์ชัน print ถ้ามีการเรียกฟังก์ชันนี้ผ่านออบเจ็คต์ของคลาสใดก็จะไปเรียกฟังก์ชันของคลาสนั้น

<u>ตัวอย่างการใช้เวอร์ชวลคลาส</u>

```
class  student :  virtual public person {
public :
   char  university[MAX] ;
   void print ( ) ;
} ;
class  employee :  virtual public person {
public :
   char  company[MAX] ;
```

```
    int  sal ;
    void print ( ) ;
} ;
class  studEmp :  public student, public employee {
public :
    int  maxhours ;
} ;
```




**StudEmp**

รูปที่ 2.3 แสดงเวอร์ชวลคลาส

คลาสหนึ่งอาจใช้เป็นพื้นฐานของหลายๆคลาส การประกาศคลาสพื้นฐานเป็นแบบเวอร์ชวลทำให้มีส่วนของคลาสพื้นฐานไม่ซ้ำกัน ตัวชี้ของเวอร์ชวลคลาสจะชี้ไปยังส่วนที่ใช้ร่วมกันดังแสดงในรูปที่ 2.3

3. <u>การจัดเก็บออบเจ็คต์และฐานข้อมูลของออบเจ็คต์</u>

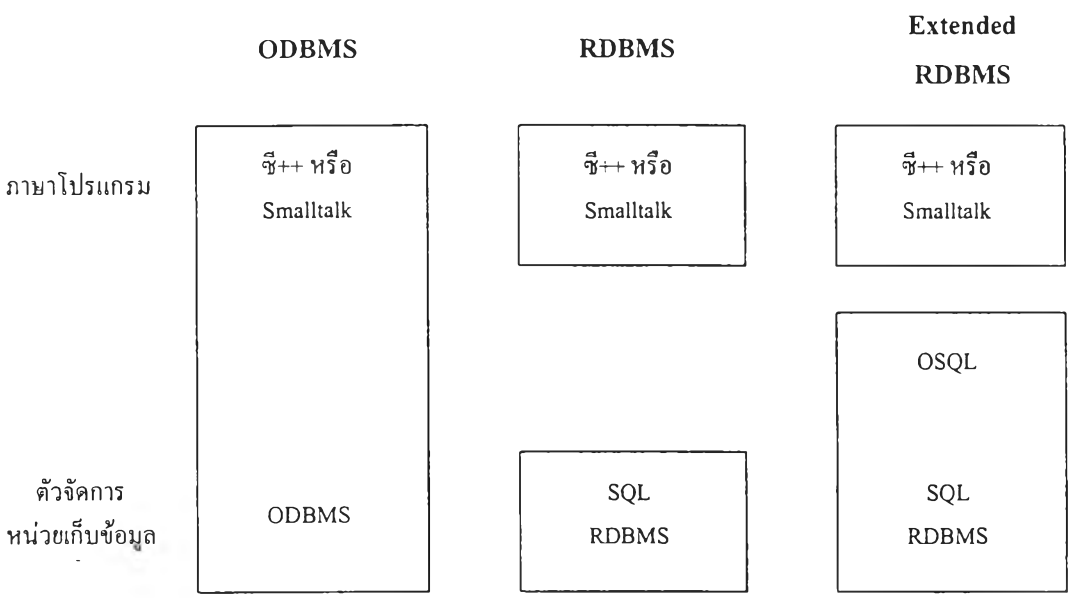
การทำให้ออบเจ็คต์คงอยู่ถาวร(persistent) ของภาษาการเขียนโปรแกรมมีอยู่หลายแนวทาง เทคนิคพื้นฐานใช้ระบบแฟ้มข้อมูล เช่น ใช้คำสั่งและฟังก์ชันของ ซี++ เพื่อเขียนออบเจ็คต์ลงแฟ้มข้อมูล วิธีนี้ต้องเขียนคำสั่งเพื่อแปลงออบเจ็คต์ในหน่วยความจำไปเป็นโครงสร้างที่สามารถเก็บในแฟ้ม แต่มีปัญหาเกี่ยวกับการอ้างอิง (references) ระหว่างออบเจ็คต์ ซึ่งเป็นตัวชี้และตำแหน่ง

ที่อยู่ในหน่วยความจำของออบเจ็คต์ ตำแหน่งของหน่วยความจำอาจไม่ถูกต้องตอนอ่านออบเจ็คต์ขึ้นมาจากแฟ้มข้อมูล ดังนั้นต้องคำนึงถึงการอ่านข้อมูลจากแฟ้มกลับมาสู่ออบเจ็คต์ นอกจากนี้ยังมีความยุ่งยากเมื่อมีการเปลี่ยนข้อกำหนดของคลาส ทำให้โครงสร้างของแฟ้มไม่ถูกต้องด้วย

ออบเจ็คต์สามารถจัดเก็บในฐานข้อมูลเชิงสัมพันธ์ (Relational Database) โดยต้องมีการแปลงจากออบเจ็คต์ไปสู่ตาราง ซึ่งคำสั่งในการแปลงนี้จะยุ่งยากซับซ้อนกว่าการใช้แฟ้มข้อมูลและต้องคำนึงถึงข้อบังคับต่างๆ ปัญหาคือชนิดของข้อมูลในแบบเชิงสัมพันธ์ไม่เพียงพอกับการจัดการข้อมูลที่ซับซ้อนของออบเจ็คต์ มีความจำกัดเรื่องค่าของข้อมูลแต่ละเซล และไม่สามารถท่องไปตามเซลด้วยตัวชี้

อีกทางเลือกหนึ่งคือใช้ฐานข้อมูลเชิงสัมพันธ์แบบขยาย (Extended Relational) ซึ่งมีส่วนเพิ่มของฐานข้อมูลเชิงสัมพันธ์ เพื่อเอื้ออำนวยกับการจัดเก็บออบเจ็คต์มากขึ้น เช่น มีออบเจ็คต์ไอเด็นติตี้ (identity) ลำดับชั้นของชนิด (type hierachy) การแปลงออบเจ็คต์ยังมีความจำเป็นเพราะหน่วยเก็บข้อมูลเป็นเชิงสัมพันธ์ แต่มีการซ่อนความสามารถการแปลงในตัว DBMS เอง

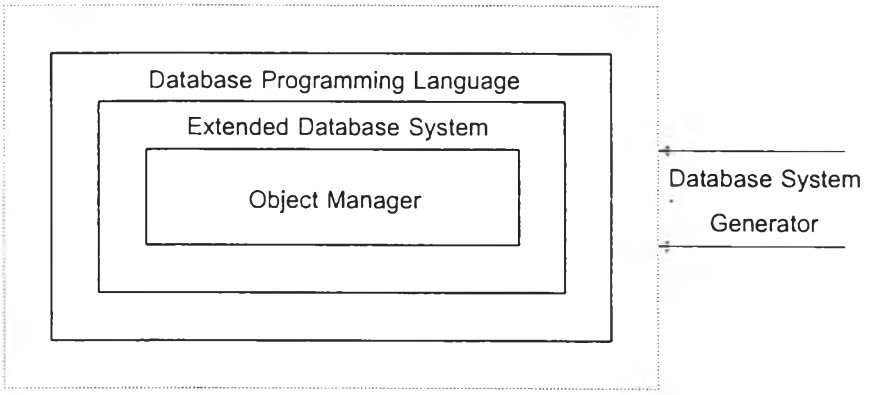
ออบเจ็คต์สามารถจัดเก็บในฐานข้อมูลของออบเจ็คต์ (Object Database) ฐานข้อมูลแบบนี้มีการรวมกับภาษาการเขียนโปรแกรมเชิงวัตถุได้ดี โมเดลของออบเจ็คต์ในหน่วยความจำและในหน่วยเก็บข้อมูลเป็นแบบเดียวกัน ผู้พัฒนาโปรแกรมไม่จำเป็นต้องเขียนคำสั่งแปลงออบเจ็คต์ในหน่วยความจำไปเป็นโครงสร้างที่ฐานข้อมูลของออบเจ็คต์จัดการได้




รูปที่ 2.4 แสดงการจัดเก็บออบเจ็คต์เพื่อการใช้งานร่วมกัน

ระบบจัดการฐานข้อมูลออบเจ็คต์ (Object Database Management) หรือ ODBMS เป็น DBMS ซึ่งเกี่ยวข้องกับโมเดลเชิงวัตถุ สามารถตอบสนองภาษาการเขียนโปรแกรมเชิงวัตถุ มีจุดประสงค์ในการจัดเก็บและใช้ออบเจ็คต์ร่วมกัน เป็นแนวทางหนึ่งสำหรับการจัดการกับหน่วยเก็บข้อมูลของออบเจ็คต์ถาวร

Cattel (1994) ได้อธิบายไว้ว่า ODBMS จะมีความแตกต่างในเรื่องของแบบจำลองข้อมูลและสถาปัตยกรรมของฐานข้อมูล ซึ่งมีสถาปัตยกรรมที่สำคัญอยู่ 4 ระดับดังแสดงในรูป 2.5




รูปที่ 2.5 แสดงความสัมพันธ์ระหว่างสถาปัตยกรรมของระบบฐานข้อมูล

1. ระบบจัดการออบเจ็คต์ (Object Managers)

ถือว่าเป็นส่วนเพิ่มของระบบแฟ้มข้อมูล มีแบบจำลองข้อมูลที่จำกัดในระดับกายภาพ จะจัดเตรียมหน่วยเก็บข้อมูลสำหรับออบเจ็คต์ถาวร แบบนี้เหมาะที่จะใช้กับโปรแกรมประยุกต์ง่ายๆ โดยมีฟังก์ชันพื้นฐานทั่วๆไปที่ใช้สำหรับทุกสถาปัตยกรรมของฐานข้อมูล จะรวมการจัดการหน่วยเก็บข้อมูล การจัดการหน่วยความจำแคช และวิธีการเข้าถึงข้อมูล (access method) อาจจะใช้เป็นระดับการจัดการหน่วยเก็บข้อมูลของระบบที่สมบูรณ์ขึ้น

2. ระบบฐานข้อมูลแบบขยาย (Extended Database Systems)

เป็นระบบฐานข้อมูลซึ่งจัดให้มีฟังก์ชันใหม่ๆ โดยสร้างภาษาการสอบถาม (query languages) ใหม่ หรือเพิ่มเติมความสามารถภาษาการสอบถามของฐานข้อมูลที่มีอยู่ให้ใช้ได้กับแนวคิดเชิงวัตถุ จะจัดเตรียมแบบจำลองข้อมูลและภาษาการสอบถามที่มีประสิทธิภาพ

3. ภาษาการเขียนโปรแกรมสำหรับฐานข้อมูล (Database Programming Languages)

ระบบฐานข้อมูลซึ่งเพิ่มจากภาษาการเขียนโปรแกรมที่มีอยู่แล้ว เช่น ซี++ เพื่อให้มีความคงอยู่ถาวร และความสามารถอื่นๆของฐานข้อมูล จะมีภาษาการสอบถามให้ใช้ เป็นการรวมกับการเข้าถึงฐานข้อมูลด้วยภาษาโปรแกรม การใช้งานมีพื้นฐานบนการแปลการทำงานของภาษาโปรแกรมเป็นการทำงานของการสอบถามในระบบฐานข้อมูลแบบขยาย

4. ตัวสร้างระบบฐานข้อมูล (Database System Generators)

ระบบซึ่งใช้สร้างหรือติดตั้ง DBMS ที่มีลักษณะพิเศษตามต้องการ สามารถใช้สร้าง ODBMS, RDBMS หรือแม้แต่ระบบจัดการแฟ้มข้อมูล โมเดลของออบเจ็คต์จะขึ้นอยู่กับการกำหนดของผู้ใช้งาน

ระบบจัดการออบเจ็คต์จะมีฟังก์ชันต่างๆน้อยกว่า ODBMS ระดับอื่นๆ โดยจัดเตรียมฟังก์ชันพื้นฐานที่สุด เช่น หน่วยเก็บข้อมูลของออบเจ็คต์ถาวร ระบบฐานข้อมูลแบบขยายจะเพิ่มความสามารถมากขึ้น รวมทั้งมีภาษาการสอบถามและการควบคุมภาวะพร้อมกันขึ้นบนระบบจัดการออบเจ็คต์ ถ้าเพิ่มการรวมกับภาษาการเขียนโปรแกรมบนระบบฐานข้อมูลแบบขยายจะเป็นภาษาการเขียนโปรแกรมสำหรับฐานข้อมูล ส่วนตัวสร้างระบบฐานข้อมูลใช้สร้างระดับต่างๆของระบบเหล่านี้

การวิจัยนี้สนใจในระดับของระบบจัดการออบเจ็คต์ ซึ่งจะจัดเตรียมความสามารถให้กับการเก็บแบบคงอยู่ถาวรของออบเจ็คต์ที่ไม่ยุ่งยากซับซ้อนนัก มีแบบจำลองข้อมูลที่ง่ายและจำกัด เช่น สำหรับโมเดลของออบเจ็คต์ซี++ มีความสามารถในการเข้าถึงออบเจ็คต์ที่เก็บไว้ เคลื่อนย้ายออบเจ็คต์ระหว่างหน่วยเก็บข้อมูลกับหน่วยความจำได้ สามารถจัดการเกี่ยวกับดิสก์ แต่ระบบจัดการออบเจ็คต์ไม่มีภาษาการสอบถาม จะใช้ภาษาการเขียนโปรแกรมในการจัดการ เช่น วิธีการของซี++ ส่วนมากไม่สามารถจัดเก็บและเข้าถึงออบเจ็คต์ในเครือข่าย (network) ตัวอย่างของระบบจัดการออบเจ็คต์ที่มีอยู่ในท้องตลาด ได้แก่ Kala เป็นผลิตภัณฑ์จากบริษัท Penobscot Development และ Mneme ซึ่งพัฒนาที่มหาวิทยาลัย Messachusetts

**4. วิธีการทำให้ออบเจ็คต์คงอยู่ถาวร**

การจัดเก็บออบเจ็คต์ลงดิสก์แล้วอ่านขึ้นมาใช้ภายหลังมีปัญหาเกี่ยวกับค่าตัวชี้ที่ซ่อนอยู่ในออบเจ็คต์ไม่ถูกต้อง ซึ่งมีแนวทางในการแก้ไขออบเจ็คต์ให้ถูกต้องอยู่หลายวิธีได้แก่

4.1. แก้ไขค่าตัวชี้ด้วยคอนสตรัคเตอร์

การแก้ไขค่าตัวชี้ด้วยคอนสตรัคเตอร์ (Biliris, Dar และ Gehani, 1993) ทำได้โดยอ่านออบเจ็คต์ที่ต้องการจากดิสก์ซึ่งค่าของตัวชี้จะไม่ถูกต้อง แล้วใช้คอนสตรัคเตอร์เพื่อเปลี่ยนค่าตัวชี้ให้ถูกต้อง คอนสตรัคเตอร์ต้องไม่ไปเปลี่ยนแปลงข้อมูลของออบเจ็คต์

วิธีนี้ต้องมีการเพิ่มคำสั่งเพื่อกำหนดค่าตัวชี้ให้คอนสตรัคเตอร์ของออบเจ็คต์แต่ละชนิด คอนสตรัคเตอร์จะถูกใช้ในทางอ้อมด้วยโอเปอเรเตอร์ new เมื่อออบเจ็คต์ถูกสร้างขึ้นมาจะมีการจัดสรรหน่วยความจำสำหรับออบเจ็คต์ แล้วทำคอนสตรัคเตอร์เพื่อกำหนดค่าเริ่มต้นของออบเจ็คต์ แต่เราไม่ต้องการเนื้อที่หน่วยความจำสำหรับออบเจ็คต์ จึงต้องมีการโอเวอร์โหลดโอเปอเรเตอร์ new

คอนสตรัคเตอร์จะแก้ไขค่าตัวชี้ให้ถูกต้องแต่ต้องไม่ไปแก้ข้อมูล จึงควรเป็นฟังก์ชันว่างๆ ตัวอย่างต่อไปนี้แสดงโอเปอเรเตอร์ new

```
class ode { };
void * operator new (size_t, ode *p)
 {  return (void *) p;
 }
new ((ode *)p) employee;
```

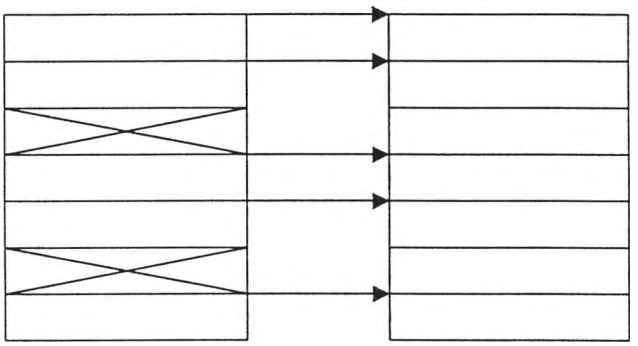
ode เป็นคลาสที่กำหนดขึ้นเพื่อโอเวอร์โหลด new p ชี้ไปยังออบเจ็คต์ที่อ่านจากดิสก์ ต้องสร้างคอนสตรัคเตอร์ว่างๆนี้สำหรับทุกคลาสโดยมีพารามิเตอร์ ode* ตัวอย่างของคลาส employee

```
employee :: employee (ode *)  { }
```

วิธีนี้ต้องมีคอนสตรัคเตอร์แบบพิเศษสำหรับแต่ละคลาส มีการกำหนดค่าคงที่และการอ้างอิง และต้องส่งค่าตำแหน่งที่อยู่ใหม่ของออบเจ็คต์กลับมาด้วย

4.2. ใช้โอเปอเรเตอร์การกำหนดค่า (assignment)

การใช้โอเปอเรเตอร์การกำหนดค่า (Biliris และคณะ, 1993) ต้องมีการจัดสรรเนื้อที่สำหรับออบเจ็คต์แล้วอ่านออบเจ็คต์ขึ้นมาจากดิสก์ ค่าของตัวชี้ที่ซ่อนอยู่ในออบเจ็คต์จะไม่ถูกต้อง จากนั้นทำการจัดสรรอีกออบเจ็คต์ ซึ่งออบเจ็คต์นี้จะมีตัวชี้ถูกต้อง แล้วกำหนดออบเจ็คต์จากดิสก์ให้กับออบเจ็คต์ใหม่ โอเปอเรเตอร์การกำหนดค่าจะกำหนดสมาชิกชนิดข้อมูลของออบเจ็คต์ต้นทางให้ออบเจ็คต์ปลายทาง โดยที่ค่าตัวชี้ไม่เปลี่ยนแปลงดังแสดงในรูปที่ 2.6 วิธีนี้สำหรับออบเจ็คต์หนึ่งหน่วยความจำจะถูกจัดสรร 2 ครั้ง และต้องรู้ว่าตัวชี้อยู่ที่ไหนบ้าง



ออบเจ็คต์ที่อ่านจากดิสก์ ออบเจ็คที่เตรียมรับข้อมูล

รูปที่ 2.6 แสดงการกำหนดค่าให้ออบเจ็คต์

4.3. ใช้ฟังก์ชันพิเศษ

การใช้ฟังก์ชันพิเศษนี้ (Soukup, 1994) แต่ละคลาสจะใช้ฟังก์ชันในการอ่านและเขียนออบเจ็คต์ การอ่านออบเจ็คต์จากดิสก์ทำโดยจัดสรรเนื้อที่สำหรับออบเจ็คต์ด้วยโอเปอเรเตอร์ new แล้วอ่านออบเจ็คต์จากดิสก์ด้วยฟังก์ชันสำหรับออบเจ็คต์ชนิดนั้น ซึ่งจะเป็นการอ่านสมาชิกข้อมูลของออบเจ็คต์ไม่ได้กระทำกับไบต์ของข้อมูล

ตัวจัดการออบเจ็คต์ไม่มีความรู้เกี่ยวกับชนิดของออบเจ็คต์ จึงต้องใช้ new ทางอ้อมด้วยฟังก์ชัน การใช้ฟังก์ชันต้องมีการกำหนดฟังก์ชันเหล่านี้เป็นสมาชิกของคลาส ออบเจ็คต์ต้องถูกอ่านจากดิสก์และถูกเขียนในลักษณะเดียวกัน วิธีการนี้สามารถทำได้หลายแบบ

แบบแรกความคงอยู่ถาวรซึ่งขึ้นกับคลาสพื้นฐาน วิธีนี้ทุกคลาสที่จะมีการจัดเก็บลงดิสก์ต้องสืบทอดมาจากคลาสๆหนึ่งซึ่งมีเวอร์ชวลฟังก์ชันสำหรับคลาสใหม่ ตัวอย่างแสดงคลาส shape ที่สามารถมีการจัดเก็บได้

```
class  shape : public Persistent {
   int  x, y ;
public :
   virtual  classID   isA ( ) const;
   virtual  void      restoreform (RWistream& );
   virtual  void      storeon ( RWostream&)  const ;
} ;
```

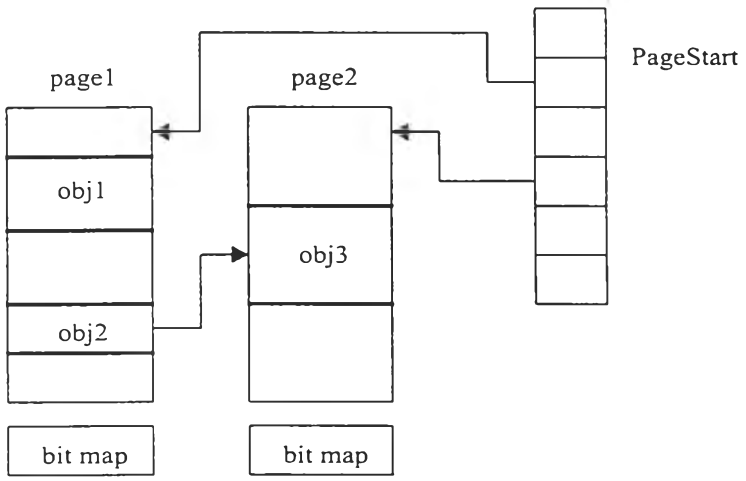
ตอนจัดเก็บต้องรู้ชนิดของออบเจ็คต์ มีการเขียน id ของคลาสที่ส่วนหัวของระเบียนแล้ว มีการเรียกใช้ฟังก์ชันที่เป็นเวอร์ชวลตามลำดับชั้น มีตารางช่วยจำว่าออบเจ็คต์ใดถูกจัดเก็บไว้แล้ว ในแฟ้มข้อมูลออบเจ็คต์จะมีหมายเลขกำกับ ตัวชี้อาจถูกแทนด้วยตัวออบเจ็คต์หรือการอ้างอิงโดยใช้หมายเลข ตอนเรียกกลับมาใช้ต้องแปลงกลับมาเป็นตัวชี้ซึ่งมีค่าใหม่ วิธีนี้ไม่ยุ่งยากนัก ต้องเตรียมฟังก์ชันสำหรับคลาสที่จะนำกลับมาใช้ และมีข้อมูลเกี่ยวกับคลาสแต่ละชนิดว่าเป็นอย่างไร มีการสืบทอดมาอย่างไร

แบบที่ 2 ความคงอยู่ถาวรที่จัดการด้วยคลาสภายนอกออบเจ็คต์จะถูกจัดเก็บด้วยยูทีลิตี้คลาส ซึ่งจัดการจัดเก็บและอ่านออบเจ็คต์โดยรู้โครงสร้างของออบเจ็คต์ทั้งหมด และสามารถจัดเก็บออบเจ็คต์ของคลาสใดก็ได้ ดังนั้นคลาสไม่ต้องสืบทอดมาจากคลาสพื้นฐานพิเศษ และไม่ต้องมี I/O ฟังก์ชันสำหรับทุกคลาส แต่ต้องบอกว่าจะจัดการกับแอททริบิวต์อย่างไร รูปแบบของแฟ้มข้อมูลเริ่มต้นด้วยตารางอธิบายชนิดของออบเจ็คต์มี 1 รายการต่อ 1 คลาสบอกชนิด ขนาด ออฟเซต จำนวนและชนิดของตัวชี้ เป็นต้น ต่อมาเป็นระเบียนของออบเจ็คต์ โดยมีส่วนหัวบอกรายละเอียดแล้วตามด้วยลิสต์ของตัวชี้และแอททริบิวต์ ตัวชี้ที่เก็บจะเป็นตัวชี้เดิมซึ่งจะมีการแปลงค่าตอนอ่านออบเจ็คต์ขึ้นมา

4.4. <u>การกระจายหน่วยความจำ (Memory Blasting)</u>

วิธีการนี้หน่วยความจำทั้งหน้า (page) ถูกจัดเก็บลงดิสก์ (Soukup, 1994) โดยไม่ต้องแบ่งเป็นออบเจ็คต์ ตอนอ่านข้อมูลจากดิสก์ทั้งหน้าก็จะถูกอ่านเข้ามาในหน่วยความจำ วิธีนี้มีข้อดีคือจะทำ I/O เร็วกว่าวิธีอื่น และไม่จำเป็นต้องมีส่วนหัวสำหรับทุกออบเจ็คต์ แต่ข้อเสียคือขึ้นอยู่กับการจัดสรรหน่วยความจำ และไม่สามารถย้ายข้อมูลระหว่างเครื่องต่างชนิดกัน

เราไม่สามารถนำหน้าของหน่วยความจำมาไว้ยังหน่วยความจำตำแหน่งเดิมได้ จึงต้องแก้ไขค่าตัวชี้ให้ถูกต้อง แนวความคิดคือให้หน่วยความจำแต่ละหน้ามีบิตแมพซึ่งจำตำแหน่งของตัวชี้ในหน้า อาจใช้ 1 บิตต่อหนึ่งตำแหน่งตัวชี้ บิตเหล่านี้ถูกกำหนดขึ้นตอนจัดสรรเนื้อที่ของออบเจ็คต์ จะถือเป็นส่วนหนึ่งของหน้าและถูกจัดเก็บลงดิสก์พร้อมกันด้วย ตอนนำกลับมาสู่หน่วยความจำบิตแมพจะถูกใช้เป็นข้อมูลว่าตัวชี้ตรงไหนที่ต้องแก้ไข




รูปที่ 2.7 แสดงโครงสร้างของข้อมูลภายในสำหรับ memory blasting

จากรูปแสดงโครงสร้างของข้อมูลภายใน ออบเจ็คต์ต้องไม่อยู่ข้ามขอบเขตของหน้า ตำแหน่งของตัวชี้ถูกบันทึกอยู่ในบิตแมพซึ่งอยู่ท้ายหน้า มีอะเรย์ซึ่งเก็บตัวชี้ที่ชี้ไปยังจุดเริ่มต้นของหน้า ตัวอย่างเช่นแต่ละหน้ามีขนาด 1024 มี 4 หน้าซึ่งเริ่มต้นที่ 12304, 13584, 15248 และ 16404 ถ้าตำแหน่ง 13628 มีตัวชี้อยู่ ดังนั้นบิต (13628 - 13584) / 4 = 11 ในบิตแมพของหน้าที่ 13 มีค่าเป็น 1

ตอนจัดเก็บข้อมูลลงดิสก์จะเอาทั้งหน้าในหน่วยความจำและบิตแมพที่เกี่ยวข้องลงดิสก์ ตอนอ่านขึ้นมาจะจัดสรรหน้าและบิตแมพใหม่แล้วเติมด้วยข้อมูลจากดิสก์ หลังจากทุกหน้าถูกนำกลับมาสู่หน่วยความจำ ก็จะดูบิตแมพซึ่งตำแหน่งของมันมีค่าเป็น 1 เพื่อแก้ไขค่าตัวชี้ แต่ในออบเจ็คต์มีตัวชี้อยู่หลายชนิดต้องการการแก้ไขแตกต่างกัน จึงอาจใช้มากกว่า 1 บิตเพื่อบอกตำแหน่งและชนิดของตัวชี้

<u>การอ่านออบเจ็คต์จากดิสก์</u>

ตอนจัดเก็บออบเจ็คต์ต้องรู้ว่าจะจัดเก็บออบเจ็คต์ใดบ้าง แล้วจึงเคลื่อนย้ายลงดิสก์ เมื่ออ่านออบเจ็คต์จากดิสก์มาไว้ในหน่วยความจำอาจถูกย้ายตำแหน่ง ต้องจัดการให้มันอยู่รวมกัน

อย่างถูกต้อง โดยตัวชี้ของเวอร์ชวลฟังก์ชันและเวอร์ชวลคลาสต้องแสดงค่าในสภาพแวดล้อมใหม่ ต้องแก้ไขตัวชี้แสดงความสัมพันธ์ให้เป็นตำแหน่งใหม่ของออบเจ็คต์ กรณีที่โครงสร้างของออบเจ็คต์ไม่มีการเปลี่ยนแปลงตอนจัดเก็บและอ่านข้อมูลขึ้นมาจะเหมือนกัน ค่าออฟเซตและชุดของตัวชี้จะเหมือนเดิม การอ่านออบเจ็คต์จากดิสก์ต้องกำหนดค่าตัวชี้ที่ซ่อนอยู่ในออบเจ็คต์ให้ถูกต้อง

ถ้าตัวชี้แสดงความสัมพันธ์ชี้ไปยังจุดเริ่มต้นของออบเจ็คต์จะแก้ไขได้ง่าย แต่บางตัวชี้ไม่อยู่ที่จุดเริ่มต้นของออบเจ็คต์ ตัวชี้อาจชี้เข้าไปภายในของออบเจ็คต์อื่น ตอนแก้ไขต้องคำนวณออฟเซตแล้วแปลงให้กับตัวชี้ใหม่

**5. <u>สิ่งที่ระบุออบเจ็คต์ (Object Identity) และคีย์ของออบเจ็คต์ (Object Key)</u>**

การใช้งานออบเจ็คต์ถาวรต้องมีการกำหนดหรือระบุออบเจ็คต์ได้ เมื่อโปรแกรมสร้างออบเจ็คต์ขึ้นมาจะใช้ตำแหน่งที่อยู่ในหน่วยความจำของออบเจ็คต์เป็นสิ่งระบุ หลังจากโปรแกรมสิ้นสุดลงไม่สามารถใช้ที่อยู่ในการระบุออบเจ็คต์ได้ต่อไป เมื่อมีการเก็บออบเจ็คต์ในฐานข้อมูลต้องมีข้อมูลในการระบุซึ่งโปรแกรมรู้จักและใช้ดึงออบเจ็คต์จากฐานข้อมูล

มีวิธีในการเข้าถึงออบเจ็คต์ในฐานข้อมูล 2 วิธีคือ

1) การเข้าถึงแบบการท่องไป (Navigational)

การระบุแยกแยะออบเจ็คต์เกี่ยวข้องกับที่อยู่ทางตรรก (logical) ในฐานข้อมูล โปรแกรมใช้ที่อยู่เหล่านี้เพื่อเรียกใช้ออบเจ็คต์

2) การเข้าถึงแบบเกี่ยวข้องกัน (Associative)

ใช้ข้อมูลที่เป็นคีย์ในการเชื่อมโยงออบเจ็คต์กับที่อยู่ในฐานข้อมูล โปรแกรมเรียกใช้ออบเจ็คต์โดยใช้ค่าของคีย์หลัก (primary key) ในการค้นหา

ออบเจ็คต์ต้องมีสิ่งที่ใช้ระบุไม่ซ้ำกัน ระบบซึ่งขึ้นอยู่กับไอเด็นติตี้ยอมให้ออบเจ็คต์ถูกอ้างอิงผ่านตัวเลขที่ไม่ซ้ำกันที่ถูกสร้างขึ้นซึ่งเรียกว่าออบเจ็คต์ไอเด็นติตี้หรือ OID ค่านี้จะไม่ขึ้นอยู่กับคีย์หลักหรือค่าของข้อมูล OID ใช้อ้างอิงออบเจ็คต์ในฐานข้อมูล เช่น ในการเกี่ยวข้องกับออบเจ็คต์อื่นหรือใช้ในโปรแกรม

OID อาจเป็นตำแหน่งที่อยู่ของออบเจ็คต์หรือแปลงดัชนีเพื่อให้ได้ที่อยู่ของออบเจ็คต์ ใช้ในการอ้างถึงออบเจ็คต์ในโปรแกรมและการอ้างอิงระหว่างออบเจ็คต์เป็นแอททริบิวต์แสดงความสัมพันธ์ ต้องมีค่าไม่ซ้ำกันทั้งฐานข้อมูล

การระบุถึงออบเจ็คต์สามารถใช้คีย์หลักและค่าของข้อมูล บาง DBMS ยอมให้ใช้ชื่อที่สื่อความหมายเป็นสิ่งระบุถึงออบเจ็คต์เรียกว่าคีย์ของออบเจ็คต์ คีย์เหล่านี้เทียบเท่ากับคีย์หลักในโมเดลเชิงสัมพันธ์ ซึ่งการใช้งานของออบเจ็คต์คีย์ก็ไม่แตกต่างจากคีย์หลัก ในระบบฐานข้อมูลที่ไม่ใช้คีย์จะสูญเสียความสามารถบางอย่างไป

การใช้คีย์สามารถใช้แอททริบิวต์ใดแอททริบิวต์หนึ่งของออบเจ็คต์เป็นคีย์ที่ใช้ค้นหาออบเจ็คต์ เช่น ใช้ชื่อของลูกจ้างหรือเลขประจำตัวเป็นคีย์ของคลาสลูกจ้าง ประโยชน์ของการมีคีย์คือ ทำให้การอ้างถึงออบเจ็คต์ทำได้ง่าย และสามารถตรวจสอบข้อบังคับ (constraint) ว่าค่าคีย์ไม่ซ้ำกัน ข้อบังคับเพื่อการไม่ซ้ำกันของคีย์อาจแตกต่างกัน บางระบบอาจต้องการค่าคีย์ไม่ซ้ำกันบนออบเจ็คต์แต่ละชนิด การใช้งานออบเจ็คต์จะต้องถูกอ้างอิงถึงโดย OID หรือคีย์ของออบเจ็คต์

**6. <u>วิธีการเข้าถึงข้อมูล (Access Methods)</u>**

Catell(1994) ได้อธิบายไว้ว่าการค้นหาข้อมูลเกี่ยวข้องกับวิธีการเข้าถึงข้อมูล ซึ่งเป็นเทคนิคเกี่ยวกับการจัดโครงสร้างของแฟ้มข้อมูลเพื่อให้ค้นหาระเบียน (record) ได้อย่างรวดเร็ว โดยให้ค่าแอททริบิวต์ที่ต้องการหรือความสัมพันธ์กับระเบียนอื่น มีวิธีการที่สำคัญ 3 ชนิดคือ ดัชนีแฮช (Hash Index) ดัชนีบี-ทรี (B-tree Index) และลิงค์แสดงความสัมพันธ์พ่อลูก (Parent-Child Link)

ทุกวิธีการเข้าถึงข้อมูลจำนวนครั้งของการอ่านดิสก์ที่ต้องใช้ในการดึงระเบียนมีความสำคัญมากกว่าปัจจัยอื่นๆ เช่น เวลาของ CPU โมเดลของออบเจ็คต์มีลักษณะเป็นลำดับชั้นของคลาส การค้นหาขึ้นอยู่กับชนิดของออบเจ็คต์

6.1 <u>ดัชนีแฮช</u>

ใช้ค้นหาระเบียนด้วยค่าของแอททริบิวต์หรือกลุ่มของแอททริบิวต์ วิธีนี้เป็นพื้นฐานที่สุดในการค้นหาระเบียนด้วยคีย์หลัก ดัชนีแฮชทำด้วยฟังก์ชันแฮชซึ่งเป็นการคำนวณทางคณิตศาสตร์

บนค่าแอททริบิวต์ซึ่งจะใช้เป็นคีย์ในการค้นหาระเบียน เพื่อหาค่าที่กระจายอยู่ในช่วง [0...N] ค่าที่ได้ใช้กำหนดตำแหน่งที่อยู่ในดิสก์ว่าระเบียนเก็บอยู่ที่ไหน

ตัวอย่างฟังก์ชันแฮชง่ายๆบนแอททริบิวต์เลขจำนวนเต็มอาจใช้บิตนัยสำคัญต่ำของค่านั้นๆ สำหรับข้อมูลจำนวนไม่เยอะเราอาจใช้ 10 บิตทำให้ค่า N = 1023 แล้วพยายามเก็บแต่ละระเบียนยังตำแหน่งที่ได้มาจากค่าฟังก์ชันแฮช ถ้ามีหลายระเบียนจะมาเก็บอยู่ที่ตำแหน่งเดียวกัน ส่วนเกินสามารถเก็บในตำแหน่งที่ว่างต่อมาหรือเก็บเป็นลิสต์ของระเบียน

ปัญหาของแฮชชิงคือ ยากในการสร้างฟังก์ชันแฮชให้กระจายอยู่ในช่วง [1..N] โดยไม่รู้ถึงค่าอินพุทที่จะเข้ามาก่อน ค่าของ N อาจต้องเพิ่มขึ้น ดังนั้นระเบียนทั้งหมดต้องถูกจัดใหม่ ถ้าแฟ้มข้อมูลมีขนาดใหญ่ขึ้นและจำเป็นต้องใช้การอ่านดิสก์หลายครั้งในการหาระเบียนกรณีเลวร้ายสุดเมื่อหลายระเบียนแฮชไปยังตำแหน่งเดียวกัน

6.2 ดัชนีบี-ทรี

ดัชนีแฮชไม่ค่อยมีประโยชน์เมื่อการค้นหาทำบนค่าเป็นช่วงของแอททริบิวต์มากกว่าเป็นค่าเดียว เมื่อต้องการเรียงลำดับระเบียน หรือเพื่อดึงข้อมูลโดยเรียงลำดับเพิ่มค่า กรณีนี้ปกติจะใช้บี-ทรี ซึ่งอาจใช้สำหรับการค้นหาแบบถูกต้องตรงกันด้วยประสิทธิภาพใกล้เคียงกับดัชนีแฮช

บี-ทรีเป็นดัชนีที่มีโครงสร้างเป็นต้นไม้ (tree) โหนดของต้นไม้บรรจุลิสต์ที่เรียงลำดับของค่าคีย์ด้วยตัวชี้ที่เกี่ยวข้อง ตัวชี้อาจอ้างอิงระเบียนด้วยค่าคีย์ที่ให้หรือโหนดอื่นในบี-ทรี เพื่อหาระเบียนที่อยู่ในช่วงระหว่างค่าคีย์ที่ให้มา การค้นหาเริ่มต้นที่รากและเสร็จสมบูรณ์เมื่อโหนดที่เป็นใบของบี-ทรีที่พบนั้นอ้างอิงระเบียนด้วยค่าคีย์ที่ให้หรืออยู่ในช่วงของค่า

6.3 ลิงค์แสดงความสัมพันธ์พ่อลูก

โครงสร้างของลิงค์ซึ่งใช้มากใน DBMS ที่ไม่เป็นเชิงสัมพันธ์ ลิงค์เป็นตัวชี้ที่ใช้เชื่อมโยงระเบียนที่สัมพันธ์กันเข้าด้วยกัน ปกติตัวชี้ที่ใช้จะเป็น ID ของระเบียน

ลิงค์พ่อลูกถูกใช้เพื่อเชื่อมโยงระเบียนซึ่งมีค่าคีย์นอก (foreign key) เดียวกันเข้ากับระเบียนที่ถูกอ้างถึงด้วยคีย์หลักที่เกี่ยวข้อง ในระดับโครงร่าง (schema) ลิงค์ถูกสร้างสำหรับความสัมพันธ์เพื่อเตรียมการเชื่อมเข้าด้วยกัน (join) ของตารางตามค่าคีย์

การแสดงลิงค์ใน DBMS ระเบียนลูก (child record) จะบรรจุฟิลด์หนึ่งด้วยตัวชี้ชี้ไปยังระเบียนพ่อ(parent record) ทำให้เราหาระเบียนพ่อได้รวดเร็ว ระเบียนที่เกี่ยวข้องกันถูกหาเจอในการอ่านดิสก์ครั้งเดียวด้วยการทำการเชื่อมต่อด้วยความสัมพันธ์แทนการอ่านดิสก์หลายครั้งด้วยดัชนี

การวิจัยนี้ใช้ลิงค์ลิสต์เพื่อเชื่อมโยงโหนดข้อมูลของออบเจ็คต์ และใช้ดัชนีบี-ทรีเพื่อเป็นดัชนีไปยังออบเจ็คต์ที่มีการจัดเก็บ

7. การออกแบบเชิงวัตถุ

Wang (1994) กล่าวถึงการออกแบบเชิงวัตถุว่า การจัดการเชิงวัตถุต้องมีวิธีการใหม่เพื่อออกแบบระบบ ในการออกแบบเชิงวัตถุปัญหาจะถูกแยกออกเป็นออบเจ็คต์ที่เกี่ยวข้องกับสิ่งที่เราสนใจในขอบเขตของปัญหา การเริ่มต้นขั้นตอนออกแบบต้องมีความเข้าใจเกี่ยวกับความต้องการและวัตถุประสงค์ของระบบที่จะสร้างขึ้น ขั้นแรกต้องวิเคราะห์ปัญหาและแบ่งออกเป็นส่วนๆที่สามารถจัดการได้แล้วกำหนดความสัมพันธ์ระหว่างกัน การแบ่งนี้ทำให้เกิดข้อกำหนดของการออกแบบ ซึ่งจะเปลี่ยนไปสู่การติดตั้งใช้งาน มีวิธีในการจัดแบ่ง (Decomposition) 3 แบบคือ

1. การแบ่งตามกระบวนการทำงาน (procedure)
2. การแบ่งตามข้อมูล
3. การแบ่งเชิงวัตถุ

ในมุมมองของการแบ่งเชิงวัตถุจะแยกระบบออกเป็นหน่วยย่อยอิสระซึ่งเกี่ยวข้องกับการกระทำต่างๆในปัญหา หน่วยย่อยเหล่านี้แทนด้วยออบเจ็คต์ซึ่งเป็นตัวแทนของสิ่งที่มีอยู่จริง มีพฤติกรรมและการตอบสนอง องค์ประกอบภายในออบเจ็คต์สนับสนุนการกระทำภายนอก การแยกแบบเชิงวัตถุนี้มีความยืดหยุ่นและใกล้เคียงกับขอบเขตของปัญหา

กระบวนการออกแบบเชิงวัตถุอาจจะเกี่ยวข้องกับการแบ่งระบบทั้ง 3 แบบ การแบ่งตามกระบวนการทำงานเน้นลำดับขั้นตอนการแก้ไขปัญหา ส่วนการแบ่งเชิงวัตถุเน้นการกระทำระหว่างสิ่งที่เราสนใจ ตอนแรกอาจใช้การมองเชิงวัตถุเพื่อกำหนดออบเจ็คต์และลักษณะพฤติกรรม แล้วลำดับเหตุการณ์เข้ากับการกระทำระหว่างออบเจ็คต์เหล่านี้ ในแง่ของการแบ่งตามข้อมูลสามารถช่วยกำหนดออบเจ็คต์ ความสัมพันธ์ ขอบเขตของการรับรู้และการเข้าถึงข้อมูล ซึ่งการแบ่งตามข้อมูลถูกใช้กับการแบ่งเชิงวัตถุด้วย การป้องกันและการปิดบังจากภายนอกใช้ทั้งโครงสร้างของข้อมูลและกระบวนการทำงานภายใน การจัดโครงสร้างเชิงวัตถุจึงเหมือนกับการจัดองค์กรของหน่วยงาน

หลักการออกแบบเชิงวัตถุ

การออกแบบเชิงวัตถุมีขั้นตอนการทำงานดังนี้

1. ระบุคลาสทั้งหมดในระบบ

2. กำหนดลักษณะพฤติกรรมภายนอกของแต่ละออบเจ็คต์

3. กำหนดข้อมูลและการทำงานภายในของออบเจ็คต์

4. ระบุการตอบสนองต่อการกระทำของออบเจ็คต์

5. ระบุบริการที่ต้องการจากออบเจ็คต์อื่นสำหรับแต่ละออบเจ็คต์

6. สร้างความสัมพันธ์ของออบเจ็คต์ต่อออบเจ็คต์อื่น

7. รวมกลุ่มออบเจ็คต์ที่คล้ายกันเข้าด้วยกันเป็นลำดับชั้นของคลาส และปรับความสัมพันธ์ของคลาส

8. ติดตั้งข้อกำหนดของการออกแบบ

การออกแบบเริ่มด้วยปัญหาและความต้องการ ต้องตรวจสอบปัญหา หาแนวทางการแก้ไข และกำหนดแบบจำลองการประมวลผลตามแนวทางนั้นที่ระดับบน ต้องมีการแตกย่อยปัญหา มีการกำหนดขอบเขตซึ่งสามารถปรับเปลี่ยนได้ แล้วพิจารณาสิ่งที่จะเป็นออบเจ็คต์ พิจารณาคำจำกัดความที่ใช้อธิบายปัญหาเป็นคลาส

เมื่อกำหนดคลาสและออบเจ็คต์ แต่ละออบเจ็คต์ถูกกำหนดลักษณะด้วยพฤติกรรมภายนอก ออบเจ็คต์มีการกระทำและตอบสนองกับการติดต่อจากภายนอกด้วยฟังก์ชันการทำงานจากภายใน การออกแบบอาจเป็นการเขียนคำอธิบายบทบาทและการกระทำของออบเจ็คต์

การออกแบบออบเจ็คต์ต้องพิจารณาฟังก์ชันการติดต่อกับภายนอก โครงสร้างของข้อมูลและการทำงานภายในของออบเจ็คต์ ออบเจ็คต์แสดงที่ระดับบนสามารถแบ่งย่อยลงมาด้วยการใช้วิธีการออกแบบเชิงวัตถุ โดยคลาสถูกกำหนดขึ้นเพื่อสนับสนุนการซ่อนและปิดบังข้อมูล ส่วนอาร์กิวเมนต์ของฟังก์ชันสะท้อนให้เห็นถึงข้อมูลที่จำเป็นสำหรับการทำงาน

สำหรับความสัมพันธ์ระหว่างออบเจ็คต์และคลาส มองภาพว่าออบเจ็คต์ในระบบทำฟังก์ชันที่ต้องการทั้งหมดอย่างไร โดยหาการตอบสนองของแต่ละออบเจ็คต์ การบริการที่ต้องการจากออบเจ็คต์อื่น และความสัมพันธ์ระหว่างออบเจ็คต์ แล้วรวมกลุ่มออบเจ็คต์ที่สัมพันธ์กันเข้าด้วยกัน รูปแบบลักษณะพฤติกรรมของออบเจ็คต์และคลาสถูกนำมาหาความสัมพันธ์

ส่วนการติดตั้งใช้งานเป็นขั้นตอนของการเขียนโปรแกรม คลาส โมดูลและออบเจ็คต์ถูกจัดทำเป็นคำสั่งโปรแกรม การทำให้ข้อกำหนดของการออกแบบกลายเป็นโปรแกรมไม่ใช่เรื่องง่ายนัก บางครั้งในขั้นตอนนี้เจอปัญหาที่มองไม่เห็นตอนออกแบบ จึงต้องมีการทบทวนในขั้นตอนการออกแบบใหม่