บทที่ 7
เทคนิคการจัดการใน PSUbase

ในบทนี้จะกล่าวถึงเทคนิคการจัดการใน PSUbase ว่ามีหลักการออกแบบการจัดการไฟล์ฐานข้อมูลแบบรีเลชัน หรือระบบ PSUbase อย่างไร และการทำงานของฟังก์ชันต่างๆ

7.1 หลักการออกแบบระบบอินเด็กซ์ไฟล์

ในหัวข้อนี้จะขออ้างถึงผลของโครงการพัฒนาระบบอินเด็กซ์ไฟล์สำหรับการจัดการฐานข้อมูล (Development of Indexed File System for DataBase Management) ซึ่งผู้เขียนได้ทำการวิจัยเมื่อปี พ.ศ. 2535 และได้จัดพิมพ์รายงานผลการวิจัย เสนอต่อสำนักงานคณะกรรม การวิจัยแห่งชาติแล้ว ซึ่งในรายงานเล่มนั้นจะกล่าวถึงหลักการออกแบบและขั้นตอนการทำงานในระบบอินเด็กซ์ไฟล์ กล่าวโดยสรุปหลักการออกแบบระบบอินเด็กซ์ไฟล์เป็นดังนี้

1. ไฟล์ที่จะใช้มี 2 ไฟล์คือ ไฟล์สำหรับเก็บข้อมูล และไฟล์สำหรับเก็บคีย์ ตัวอย่างเช่น ไฟล์ชื่อ emp ใช้สำหรับเก็บข้อมูล และไฟล์ emp.inx ใช้สำหรับเก็บคีย์ เป็นต้น

2. การจัดเก็บคีย์ในไฟล์คีย์ จะมีโครงสร้างแบบต้นไม้ (Tree Structure)

3. ความยาวของเร็คคอร์ดในไฟล์ข้อมูลจะเป็นแบบความยาวคงที่ (Fixed length)

4. การจัดการกับข้อมูล เช่นการอ่าน (Read) การเขียน(Write) และการแก้ไข (Rewrite) จะกระทำโดยใช้คีย์ค้นหาในไฟล์เก็บคีย์ก่อน เพื่อให้ทราบว่ามีข้อมูลที่ต้องการหรือไม่ หลังจากนั้นก็จะไปทำงานต่อที่ไฟล์เก็บข้อมูล

รายละเอียดเพิ่มเติมเรื่องนี้ สามารถศึกษาได้จากรายงานการวิจัยเรื่องการพัฒนาระบบอินเด็กซ์ไฟล์ สำหรับการจัดการ ฐานข้อมูล

7.2 หลักการออกแบบระบบ PSUbase

จากผลที่ได้จากการวิจัยเรื่องการพัฒนาระบบอินเด็กซ์ไฟล์ สำหรับการจัดการฐานข้อมูล ผู้เขียนจึงมีแนวทางที่จะทำการวิจัยต่อไปอีก โดยมีหลักการดังนี้

1. การกระทำกับข้อมูลควรจะกระทำในระดับของฟิลด์ (Fields Operation) กล่าวคือจากระบบอินเด็กซ์ไฟล์เดิมที่ใช้ไฟล์ 2 ไฟล์ ในการจัดการกับข้อมูลนั้น พบว่ายังไม่ค่อยเหมาะสำหรับผู้ใช้งานทั่วๆไป โดยเฉพาะเรื่องการปรับปรุงโครงสร้างไฟล์จะทำไม่ได้ ดังนั้นจึงกำหนด

ให้มีไฟล์ขึ้นอีก 1 ไฟล์ คือไฟล์นามสกุล stc ซึ่งใช้อธิบายโครงสร้างของไฟล์ว่าประกอบไปด้วยฟิลด์อะไรบ้าง ฟิลด์ใดใช้เป็นคีย์ ความยาวของฟิลด์ และเป็นฟิลด์ชนิดตัวอักษรหรือตัวเลข ซึ่งชื่อฟิลด์เหล่านี้จะนำไปใช้ประโยชน์ใน PSU*SQL, PSU*FORM และ PSU*ProC

2. สร้างฟังก์ชันแบบ Macro ทำหน้าที่จัดการด้าน Input/Output จากฟังก์ชัน I/O มาตรฐานในภาษาซี เช่น ฟังก์ชัน fopen(), fgets() และ fputs() เป็นต้น ซึ่งการกระทำกับข้อมูลจะเป็นแบบเรียงลำดับเร็คคอร์ด (sequential) ซึ่งไม่เหมาะกับงานด้านฐานข้อมูล ดังนั้นผลการวิจัยเรื่องแรกจึงสามารถจัดการกับไฟล์ข้อมูลแบบอินเด็กซ์ไฟล์ได้ กล่าวคือได้สร้างฟังก์ชัน เช่น fopeninx(), freadinx() และ frewriteinx() ให้ช่วยจัดการกับไฟล์แบบอินเด็กซ์ไฟล์ แต่ก็ยังไม่มีความคล่องตัวเท่าที่ควร ดังนั้นโครงการวิจัยนี้จึงได้กำหนดให้มีฟังก์ชันแบบ Macro ให้จัดการกับอินเด็กซ์ไฟล์ในระดับที่สูงขึ้น เช่นฟังก์ชัน select(), use(), skip(), และ seekdbf() เป็นต้น ซึ่งจะทำให้การเขียนโปรแกรมใน PSU*ProC กระทัดรัดขึ้น ซึ่งการเขียนโปรแกรมจะใกล้เคียงกับการเขียนบน FoxPro ตัวอย่างเช่น ต้องการอ่านไฟล์ emp แล้วนำรหัส dept_no ไปค้นหาชื่อ dept_name ในไฟล์ dept สามารถเขียนเป็นโปรแกรมได้ดังนี้

เปรียบเทียบการเขียนโปรแกรมระหว่าง FoxPro กับ PSU*ProC

FoxPro	PSU*ProC

```
PROCEDURE EXAM
SELECT 1
USE EMP
SELECT 2
USE DEPT INDEX DEPT
SELECT 1
DO WHILE .NOT. EOF()
  SELECT 2
  SEEK A->DEPT_NO
  ? emp_id,emp_name,dept_name
  SELECT 1
  SKIP
ENDDO
CLOSE DATA
```

```
main()
{
   select(1);
   use("emp","r");
   map_A();
   gotop();
   select(2);
   use("dept","r");
   map_B();
   gotop();
   select(1);
   while (!eof()) {
     select(2);
     seekdbf(A_dept_no);
     printf("%s %s %s\n",A_emp_id,
            A_emp_name,B_dept_name);
     select(1);
     skip(1);
   }
   closedata();
}
```

รายละเอียดของ PSU*ProC ดูได้จากไฟล์ sam1.c ในหัวข้อที่ 6.3 หน้าที่ 41จาก ตัวอย่าง จะเห็นได้ว่า การเขียนโปรแกรมภาษาซีจะกระทัดรัดขึ้น ซึ่งน่าจะเป็นแนวทางหนึ่งในการใช้ภาษาซีในงานฐานข้อมูล

7.3 รายละเอียดใน Libery ชื่อ libdbf.a

จากหลักการออกแบบระบบ PSUbase จึงได้กำหนดให้มี libery จำนวน 2 libery คือ libery ชื่อ libinx.a และ libdbf.a โดยที่แต่ละ libery จะมีฟังก์ชันดังต่อไปนี้

libinx.a เป็น libery ที่ถูกออกแบบไว้ในโครงการวิจัย เรื่องการพัฒนาระบบอินเด็กซ์ไฟล์ ซึ่งจะมีฟังก์ชันหลักๆ คือ fopeninx(), fcloseinx(), fcreatinx(), freadinx(), fwriteinx(), และ frewriteinx() ซึ่งรายละเอียดดูได้จากรายงานการวิจัยดังกล่าว

libdbf.a เป็น libery ที่ถูกออกเพิ่มเติมต่อจาก libinx.a โดยจะทำหน้าที่จัดการฐานข้อมูล ในระดับฟิลด์ (Field operation) ในที่นี้ได้กำหนดให้มีไฟล์เก็บตัวโปรแกรมต้นฉบับ (source code) จำนวน 5 ไฟล์ ซึ่งแบ่งออกตามหน้าที่ดังนี้

ไฟล์ ut_db1.c เก็บฟังก์ชันกลุ่มที่ 1 เช่น select(), use(), mapchar() และ mapreal()

ไฟล์ ut_db2.c เก็บฟังก์ชันกลุ่มที่ 2 เช่น initscr(), endscr() และ eject()

ไฟล์ ut_db3.c เก็บฟังก์ชันกลุ่มที่ 3 เช่น defchar(), defreal() และ sayget()

ไฟล์ ut_menu.c เก็บฟังก์ชันเกี่ยวกับการทำเมนู เช่น menu_bar() และ menu_to()

ไฟล์ ut_tty.c เก็บฟังก์ชันเกี่ยวกับการใช้ tty เช่น gettty() และ whoami()

7.3.1 ไฟล์ ut_db1.c

ไฟล์ ut_db1.c ใช้สำหรับเก็บ source code ของฟังก์ชันกลุ่มที่ 1

ไฟล์ cst_db.h ใช้สำหรับเก็บค่าคงที่ต่างๆ เช่น

MAXFILE	64	หมายถึงจำนวนไฟล์ที่เปิดได้สูงสุด 64 ไฟล์
MAXFIELD	256	หมายจำนวนฟิลด์มีสูงสุดใน 1 ไฟล์
MAX1025	1025	หมายกำหนดค่าสำหรับ buffer ประมาณ 1024 bytes
MAXPULL	256	หมายจำนวนฟิลด์สูงสุดใน PSU*FORM

นอกจากนั้นยังมีค่าคงที่อื่นๆ อีก เช่น ESC = 0X1B เป็นต้น

การกำหนดตัวแปรชนิด static

จากหลักการที่จะจัดการกับไฟล์ PSUbase ให้มีความคล่องตัวมากที่สุด ดังนั้นจึงได้กำหนดให้มีฟังก์ชันแบบ macro ในการจัดการกับไฟล์ กล่าวคือใน ut_db1.c จะมีการกำหนดตัวแปรชนิด static เพื่อให้โปรแกรมเมอร์สามารถเขียนโปรแกรมด้วย PSU*ProC เพื่อทำการติดต่อกับไฟล์ PSUbase ได้

ชื่อฟังก์ชันและค่าคงที่

1. **ฟังก์ชัน** select(n) ทำหน้าที่กำหนดพื้นที่ (Working Area) สำหรับเก็บข้อมูลแต่ละเร็คคอร์ด ของไฟล์ PSUbase

พารามิเตอร์

n คือหมายเลขพื้นที่ มีค่าตั้งแต่ 1 ถึง 64 (MAXFILE)

ตัวอย่างเช่น

```
select(1);
```

2. **ฟังก์ชัน** use(fn,mode) ทำหน้าที่เปิดไฟล์ชนิด PSUbase ฟังก์ชันนี้ต้องใช้หลังจาก ใช้ฟังก์ชัน select(n)

พารามิเตอร์

fn คือชื่อไฟล์ที่จะเปิด เช่น fn="employee"

mode คือเงื่อนไขการเปิดไฟล์ ซึ่งมีค่าดังนี้

mode="r" คืออ่านได้อย่างเดียว

mode="w" คือเขียนได้อย่างเดียว

mode="r+" คืออ่านและเขียนได้

ตัวอย่างเช่น

```
select(1);
select("employee","r")
```

จะทำการเปิดไฟล์ employee ให้ใช้งานได้ต่อไป

ในกรณีที่ต้องการปิดไฟล์ ให้ใช้ฟังก์ชันดังนี้ select(" ","r")

3. **ฟังก์ชัน** use2(fn,mode,fnstc) ทำหน้าที่เปิดเหมือนกับ use(fn,mode) แต่ต่างกันตรงที่สามารถกำหนดชื่อไฟล์ fnstc ได้

4. **ฟังก์ชัน** mapchar(field_name,field_pointer) ทำหน้า mapping ชื่อฟิลด์เข้ากับ ที่อยู่ (address) ของตัวแปรชนิดตัวอักษร (character) ในภาษาซี

พารามิเตอร์

field_name คือชื่อฟิลด์ของ PSUbase ไฟล์

field_pointer คือ address ของตัวแปรชนิดตัวอักษรในภาษาซี

ตัวอย่างเช่น

```
char *emp_id;
mapchar("EMP_ID",&emp_id);
```

emp_id เป็นชื่อตัวแปรในภาษาซี

EMP_ID เป็นชื่อไฟล์ในไฟล์ employee

5. **ฟังก์ชัน** mapreal(field_name,field_pointer) ทำหน้า mapping ชื่อฟิลด์เข้ากับ ที่อยู่ (address) ของตัวแปรชนิดตัวเลข (real) ในภาษาซี

พารามิเตอร์

field_name คือชื่อฟิลด์ของ PSUbase ไฟล์

field_pointer คือ address ของตัวแปรชนิดตัวเลขในภาษาซี

ตัวอย่างเช่น

```
double salary;
mapreal("SALARY",&salary);
```

6. **ฟังก์ชัน** gotop() ทำหน้าเปลี่ยนตำแหน่งไปอยู่เร็คคอร์ดแรกของไฟล์

หลังจาก mapping ฟิลด์ครบแล้ว จะต้องใช้คำสั่ง gotop() เพื่อให้ข้อมูล เร็คคอร์ดแรกเข้ามาสู่ชื่อตัวแปรในภาษาซี

7. **ฟังก์ชัน** gobuttom() ทำหน้าเปลี่ยนตำแหน่งไปอยู่เร็คคอร์ดสุดท้ายของไฟล์

8. **ฟังก์ชัน** go(n) ทำหน้าเปลี่ยนตำแหน่งไปอยู่เร็คคอร์ดที่ n ของไฟล์

พารามิเตอร์

n คือตัวเลขเร็คคอร์ดที่จะไปอยู่

ตัวอย่างเช่น

```
go(15);
```

9. **ฟังก์ชัน** skip(n) ทำหน้าเปลี่ยนตำแหน่งเร็คคอร์ดของข้อมูล

พารามิเตอร์

n คือตัวเลขจำนวนเต็มบวกหรือลบ

ตัวอย่างเช่น

```
skip(1); หรือ
skip(-1);
```

10. **ฟังก์ชัน** seekdbf(sk) ใช้สำหรับอ่านข้อมูลแบบ Random โดยใช้ sk เป็นคีย์

พารามิเตอร์

sk คือตัวแปรชนิด string สำหรับเก็บค่าคีย์ ในการอ่านไฟล์

ตัวอย่างเช่น

```
seekdbf("1001");
```

11. **ฟังก์ชัน** found() ใช้สำหรับสอบถามว่าอ่านข้อมูล (seekdbf) พบหรือไม่

ตัวอย่างเช่น

```
seekdbf("1001");
if (found())
  statment;
```

12. **ฟังก์ชัน** repchar(field_name,field_pointer) ใช้สำหรับแทนที่ข้อมูลจากตัวแปรชนิดตัวอักษรในภาษาซี ลงฟิลด์ในหน่วยความจำ

พารามิเตอร์

field_name คือชื่อฟิลด์ของ PSUbase ไฟล์

field_pointer คือ address ของตัวแปรชนิดตัวอักษรในภาษาซี

ตัวอย่างเช่น

```
char *emp_name;
repreal("EMP_NAME",&emp_name);
```

13. **ฟังก์ชัน** repreal(field_name,field_pointer) ใช้สำหรับแทนที่ข้อมูลจากตัวแปรชนิดตัวเลขในภาษาซี ลงฟิลด์ในหน่วยความจำ

พารามิเตอร์

field_name คือชื่อฟิลด์ของ PSUbase ไฟล์

field_pointer คือ address ของตัวแปรชนิดตัวเลขในภาษาซี

ตัวอย่างเช่น

```
double salary;
repreal("SALARY",&salary);
```

14. **ฟังก์ชัน** replace() ใช้สำหรับนำข้อมูลจากฟิลด์ในหน่วยความจำลงไฟล์ PSUbase กล่าวคือเป็นการเขียนข้อมูลลงไฟล์จริงๆ คำสั่งนี้จะใช้หลังจาก repchar() และ repreal()

ตัวอย่างเช่น

```
double salary;
repreal("SALARY",&salary);
replace();
```

15. **ฟังก์ชัน** delete() ใช้สำหรับลบข้อมูลตำแหน่งเร็คคอร์ดปัจจุบัน (current record) ในการนี้จะทำการมาร์คเพื่อลบเท่านั้น (ยังไม่ลบจริงๆ) สามารถเรียกกลับมาได้ ตัวอย่างเช่น

```
double salary;
delete();
```

16. **ฟังก์ชัน** recall() ใช้สำหรับเรียกเร็คคอร์ดที่ถูกมาร์คเพื่อลบกลับมา ตัวอย่างเช่น

```
double salary;
if (isdelete())
  recall();
```

17. **ฟังก์ชัน** isdelete() ใช้สำหรับถามว่าเร็คคอร์ดนั้นถูกมาร์คเพื่อลบหรือไม่ ตัวอย่างเช่น

```
double salary;
if (isdelete())
  recall();
```

18. **ฟังก์ชัน** append() ใช้สำหรับเพิ่มเร็คคอร์ดว่างต่อท้ายไฟล์ (append blank) ตัวอย่างเช่น

```
char *skey;
seekdbf(skey);
if (! found())
  append();
```

19. **ฟังก์ชัน** eof() ใช้สำหรับสอบถามว่าขณะนี้อยู่ที่ตำแหน่ง end of file หรือไม่ ตัวอย่างเช่น

```
char *skey;
while (! eof())
  skip(1);
```

20. **ฟังก์ชัน** bof() ใช้สำหรับสอบถามว่าขณะนี้อยู่ที่ตำแหน่ง begin of file หรือไม่ ตัวอย่างเช่น

```
char *skey;
if (bof())
  printf("Now, stay at begin of file\n");
```

7.3.2 ไฟล์ ut_db2.c

ไฟล์ ut_db2.c ใช้สำหรับเก็บ source code ของฟังก์ชันกลุ่มที่ 2 ซึ่งมีดังต่อไปนี้

1. **ฟังก์ชัน** initscr() ทำหน้าที่เปลี่ยนโหมดจอภาพเป็น raw mode และ none echo โดยใช้คำสั่งดังนี้

```
system("stty raw -echo");
```

เมื่ออยู่ในโหมดนี้แล้วจะสามารถโต้ตอบกับ Host แบบไม่มี buffer บนแป้นพิมพ์

2. **ฟังก์ชัน** endscr() ทำหน้าที่เปลี่ยนโหมดจอภาพกลับสู่สภาวะปกติ โดยใช้คำสั่งดังนี้

```
system("stty -raw echo pass8");
```

นอกจากนี้ยังมีฟังก์ชันอื่นๆอีก เช่น

eject() ใช้สำหรับ พิมพ์ขึ้นหน้าใหม่

clear() ใช้สำหรับ clear จอภาพ

beep() ใช้สำหรับ แสดงเสียงบีบ

getnc() ใช้สำหรับ รอรับข้อมูลตัวเลข

text() ใช้สำหรับ แสดงข้อความบนจอภาพ

say() ใช้สำหรับ แสดงข้อความบนจอภาพในตำแหน่งที่ระบุไว้

goyx() ใช้สำหรับ ให้ cursor ไปอยู่ตำแหน่งที่ระบุไว้

7.3.3 ไฟล์ ut_db3.c

ไฟล์ ut_db3.c ใช้สำหรับเก็บ source code ของฟังก์ชันกลุ่มที่ 3 ซึ่งมีดังต่อไปนี้

1. **ฟังก์ชัน defchar()** ทำหน้าที่กำหนดตัวแปรชนิดตัวอักษรที่จะใช้ร่วมกับPSUbase กล่าวคือจะเป็นตัวแปรชนิดหน่วยความจำที่ใช้งานควบคู่กับตัวแปรของฟิลด์ต่างๆ เช่น

```
char  *emp_id;          เป็นตัวแปรของฟิลด์ชื่อ emp_id
char  *memp_id;         เป็นตัวแปรหน่วความจำของฟิลด์ชื่อ emp_id
defchar("memp_id",&memp_id);
```

2. **ฟังก์ชัน defreal()** ทำหน้าที่กำหนดตัวแปรชนิดตัวเลขที่จะใช้ร่วมกับ PSUbase ทำนองเดียวกับ defchar() ตัวอย่าง เช่น

```
double  salary;         เป็นตัวแปรของฟิลด์ชื่อ salary
double  msalary;        เป็นตัวแปรหน่วความจำของฟิลด์ชื่อ salary
defreal("msalary",&msalary);
```

3. **ฟังก์ชัน sayget()** ทำหน้าที่แสดงข้อความบนจอภาพและรอรับข้อมูลทางแป้นพิมพ์ ทำนองเดียวกับ @ say ... get ในโปรแกรม dBASE III ตัวอย่าง เช่น

```
sayget("\
  @ 01,20 say 'Emp_id   :' get memp_id  pict '99999' \
  @ 03,20 say 'Salary   :' get msalary   pict '99999.99'  rang 1,10 \
");
```

การออกแบบฟังก์ชัน sayget() จะใช้หลักการคล้ายๆ การทำงานของคอมไพล์เลอร์ คือจะมีการแปลความหมายของข้อความภายในคำสั่ง sayget() เช่น ไวยกรณ์ถูกต้อง หรือไม่ และที่สำคัญจะต้องเชื่อมโยงตัวแปรให้ได้ เช่น get memp_id

ฟังก์ชัน sayget() จะประกอบด้วยฟังก์ชันอื่นๆ เช่น

genfield() ทำหน้าที่แบ่งแยกข้อความในคำสั่ง sayget()

genyx() ทำหน้าที่ค้นหาข้อมูลตำแหน่ง YX บนจอภาพ เช่น @ 01,20

gensh() ทำหน้าที่ค้นหาข้อความแสดงผล เช่น 'Emp_id :'

genvar() ทำหน้าที่ค้นหาชื่อตัวแปรในคำสั่ง sayget() เช่น memp_id

genrang() ทำหน้าที่ค้นหาการกำหนดช่วง (rang) เช่น rang 1,10

genvalid() ทำหน้าที่ค้นหาการค่าที่ยอมรับ (valid) เช่น valid 'YyNn'

ดูรายละเอียดวิธีการเขียนโปรแกรมเรียกใช้ sayget() จากไฟล์ sam7.c หน้าที่ 46

7.3.4 ไฟล์ ut_menu.c

ไฟล์ ut_menu.c ใช้สำหรับเก็บ source code ของฟังก์ชันใช้ทำเมนู

ตัวอย่าง คำสั่ง menu_bar ทำหน้าที่เกี่ยวกับเมนูบาร์ ซึ่งใช้ใน shell script เช่น

```
#! /bin/csh
menu_bar "\
        @ 06,40 say '1. Enter Data' \
        @ 08,40 say '2. Compute Data' \
        @ 10,40 say '3. Print Report' \
"
```

หลักการออกแบบคำสั่ง menu_bar ใช้หลักการทำนองเดียวกับ sayget()

และฟังก์ชันย่อยๆในไฟล์ ut_menu.c จะใช้ชื่อขึ้นต้นด้วยอักษร 'm' เช่น

mgenfield() ทำหน้าที่แบ่งแยกข้อความในคำสั่ง menu_bar()

mgenyx() ทำหน้าที่ค้นหาข้อมูลตำแหน่ง YX บนจอภาพ เช่น @ 06,40

mgensh() ทำหน้าที่ค้นหาข้อความแสดงผล เช่น '1. Enter Data'

แต่จะแตกต่างกับ sayget() ตรงที่ไม่มีการกำหนดตัวแปร เช่น genvar()

7.3.5 ไฟล์ ut_tty.c

ไฟล์ ut_tty.c ใช้สำหรับเก็บ source code ของฟังก์ชันเครื่องมือทั่วๆไป