บทที่ 4

การออกแบบระบบ

จากการศึกษาลักษณะแฟ้มข้อมูลบนระบบยูนิกซ์ ลักษณะการรับ-ส่งข้อมูลโดย TCP/IP บนระบบยูนิกซ์ แนวคิดในการพัฒนาคลังชุดคำสั่งฐานข้อมูลของ Richard Stevens และการออกแบบ Database Engine ของ David Hughes จึงทำให้การออกแบบเพื่อการพัฒนาเครื่องมือซอฟต์แวร์สำหรับการเข้าถึงแฟ้มข้อมูลบนระบบยูนิกซ์หลายระบบ มีรายละเอียดดังต่อไปนี้

4.1 ออกแบบโครงสร้างระบบ

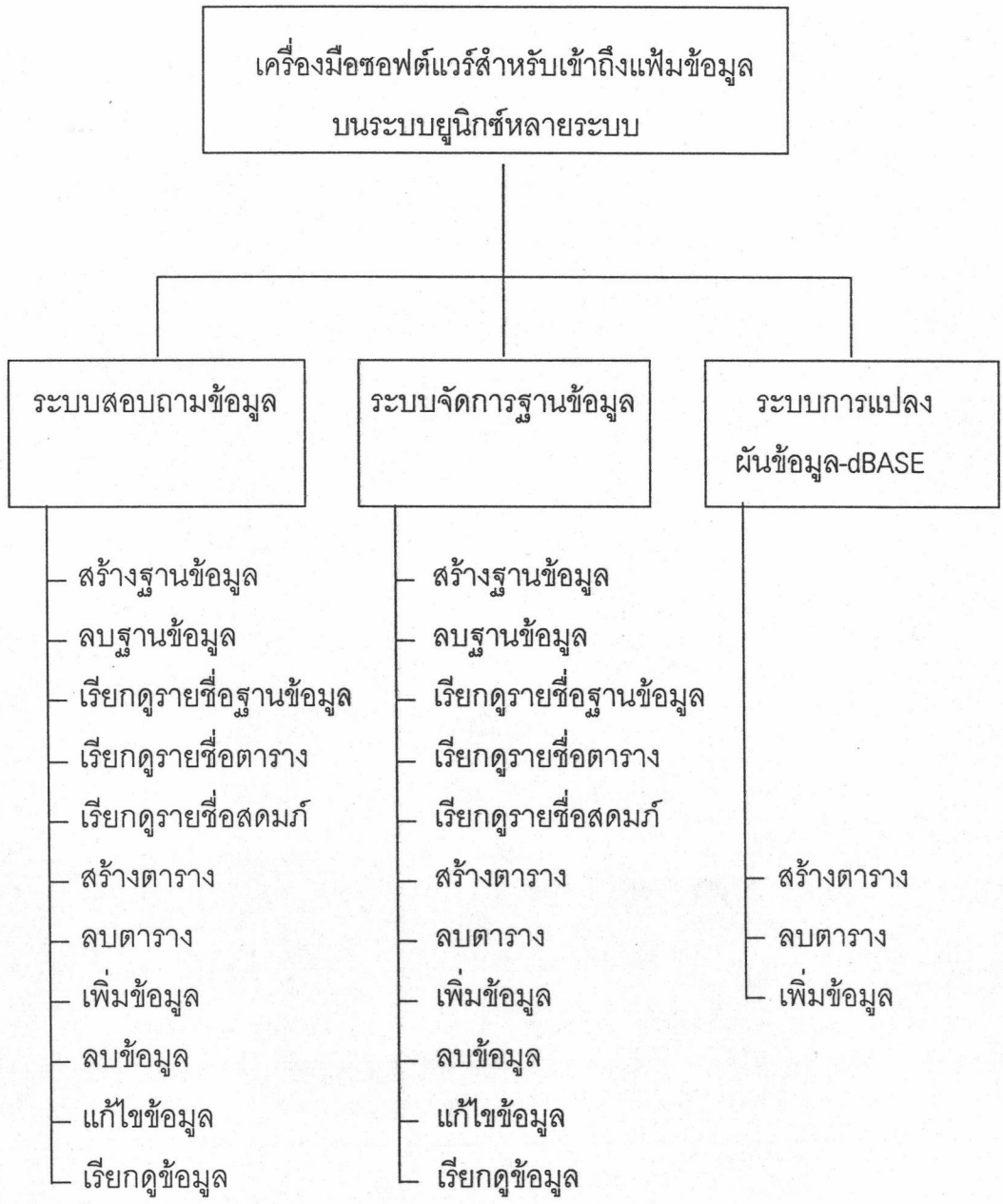
จากจุดประสงค์ที่ต้องการใช้ข้อมูลร่วมกันบนระบบยูนิกซ์ แต่เนื่องจากระบบจัดการฐานข้อมูลจะมีประสิทธิภาพมากกว่าระบบจัดการของแฟ้มข้อมูล ผู้วิจัยจึงได้ออกแบบระบบจัดการแฟ้มข้อมูลบนยูนิกซ์ให้มีคุณลักษณะใกล้เคียงกับระบบจัดการฐานข้อมูล ดังนั้นเพื่อให้ความเข้าใจตรงกันผู้วิจัยจะเรียกกลุ่มของแฟ้มข้อมูลบนยูนิกซ์ที่เป็นข้อมูลเดียวกันว่าตารางสำหรับการพัฒนาเครื่องมือซอฟต์แวร์สำหรับเข้าถึงแฟ้มข้อมูลบนระบบยูนิกซ์หลายระบบ เป็นโปรแกรมประเภท “ส่วนชุดคำสั่งระบบ (System Software)” ที่ใช้ในการบริหารฐานข้อมูล ซึ่งผู้วิจัยได้เลือกรูปแบบการทำงานของฐานข้อมูลเป็นแบบรวมศูนย์ ทั้งนี้เนื่องจาก

1) ต้องการให้มีโปรแกรมจัดการฐานข้อมูลซึ่งทำหน้าที่ควบคุมการติดต่อฐานข้อมูลเพียงกระบวนการเดียวเป็น “ผู้ให้บริการ” แก่โปรแกรมสอบถามข้อมูลที่เป็น “ผู้ใช้” โดยผู้ใช้ไม่จำเป็นต้องไปเขียนหรือแก้ไขโปรแกรมจัดการฐานข้อมูล เพียงแต่พัฒนาโปรแกรมสอบถามข้อมูลขึ้นมาใช้เองตามมาตรฐานของโปรแกรมจัดการฐานข้อมูล หรือใช้จากที่ผู้วิจัยพัฒนาขึ้น

2) ทำให้สะดวกในการเพิ่มเติมชุดคำสั่งที่ใช้ในการติดต่อบนเครือข่าย

3) ทำให้สะดวกในการควบคุมกระบวนการที่เรียกใช้ฐานข้อมูลบนเครือข่าย

จากเหตุผลที่ได้กล่าวมาข้างต้นผู้วิจัยจึงทำการออกแบบโครงสร้างระบบแยกตามหน้าที่ ดังแสดงในรูปที่ 4.1

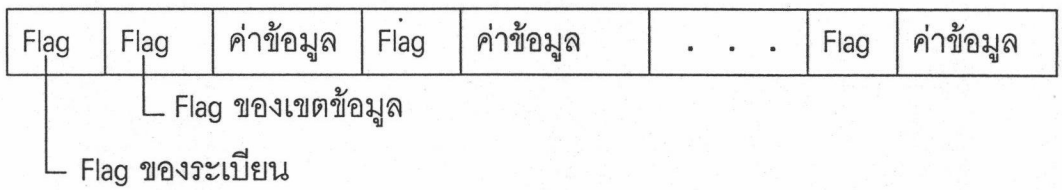


รูปที่ 4.1 โครงสร้างระบบเครื่องมือซอฟต์แวร์สำหรับเข้าถึงแฟ้มข้อมูลบนระบบยูนิกซ์หลายระบบ

4.2 โครงสร้างแฟ้มข้อมูล

ผู้ใช้งานจะมองข้อมูลที่เก็บอยู่ที่ส่วนจัดการฐานข้อมูลในรูปตาราง (Table) แต่ส่วนจัดการฐานข้อมูลจะมองตารางออกเป็น 4 แฟ้มข้อมูล โดยแต่ละแฟ้มข้อมูลมีวัตถุประสงค์และโครงสร้างข้อมูลดังนี้

4.2.1 แฟ้มข้อมูล จะมีวัตถุประสงค์สำหรับจัดเก็บข้อมูลของตาราง โดยมีโครงสร้างคือ



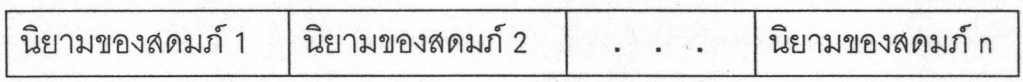
Flag ของระเบียน จะเก็บสถานะของระเบียน

- Active ข้อมูลยังใช้อยู่
- Non-active ข้อมูลถูกลบแล้ว

Flag ของเขตข้อมูล จะเก็บสถานะของเขตข้อมูล

- Null ไม่มีค่าของเขตข้อมูล

4.2.2 แฟ้มนิยาม จะมีวัตถุประสงค์สำหรับจัดเก็บข้อมูลนิยามของแต่ละสดมภ์ในตาราง โดยมีโครงสร้างคือ



นิยามของสดมภ์ จะมีขนาด 64 ไบต์ใช้เก็บข้อมูลชื่อตาราง ชื่อสดมภ์ ชนิดของข้อมูล ความยาวของสดมภ์ ค่าของ Null Flag และค่าของ Primary Key Flag

4.2.3 แฟ้มกองซ้อน จะมีวัตถุประสงค์สำหรับจัดเก็บตำแหน่งของข้อมูลที่ถูกลบ เพื่อที่จะใช้อ้างอิงเมื่อเพิ่มข้อมูลเข้าไปในตาราง ซึ่งจะสามารถลดปัญหาของการแตกกระจายโดยมีโครงสร้างดังนี้

หมายเลข row ที่ถูกลบ

หมายเลข row ที่ถูกลบ จะเป็นเลขจำนวนเต็ม (Integer)

4.2.4 แฟ้มกุญแจหลัก จะมีวัตถุประสงค์สำหรับจัดเก็บกุญแจหลักของข้อมูลในตาราง เพื่อที่จะใช้ในการค้นหาข้อมูล

Flag	ค่าข้อมูลที่เป็นกุญแจหลัก

Flag ของแฟ้มกุญแจหลัก จะเก็บสถานะของระเบียนของแฟ้มกุญแจหลัก

- Active ข้อมูลยังใช้อยู่
- Non-active ข้อมูลถูกลบแล้ว

4.3 การออกแบบระบบสอบถามข้อมูล

ระบบสอบถามข้อมูล มีหน้าที่ในการเชื่อมต่อระหว่างผู้ใช้งานกับส่วนจัดการฐานข้อมูล ส่วนนี้จะแยกเป็น 2 ระบบ คือ ระบบสำหรับผู้บริหารฐานข้อมูล และระบบสำหรับผู้ใช้บริการ ทั้งนี้เนื่องจาก

- ผู้บริหารฐานข้อมูลมีวัตถุประสงค์ในการติดต่อส่วนจัดการฐานข้อมูลเพื่อควบคุมและจัดการบำรุงรักษาฐานข้อมูล แต่ผู้ใช้บริการจะมีวัตถุประสงค์เพื่อเรียกใช้ฐานข้อมูล
- การแยกระดับของการจัดการจะทำให้ระบบสอบถามข้อมูลสามารถกำหนดสิทธิการใช้งานระบบ จากส่วนควบคุมการเข้าถึงแฟ้มข้อมูลของยูนิกซ์ (File Permission) ได้

4.3.1 การออกแบบระบบสำหรับผู้บริหารฐานข้อมูล

หน้าที่สำคัญของผู้บริหารฐานข้อมูลมีดังนี้

- ติดต่อกับผู้ใช้
- ควบคุมและจัดการบำรุงรักษาฐานข้อมูล
- ออกแบบแฟ้มต่างๆของฐานข้อมูล
- ออกแบบและจัดระเบียบการรักษาความปลอดภัย

ซึ่งรูปแบบการทำงานของผู้บริหารฐานข้อมูลจะเป็นรูปแบบที่ไม่มีความซับซ้อนของคำสั่งในการดูแลจัดการฐานข้อมูลต่างๆ เมื่อพิจารณาเทียบความซับซ้อนของคำสั่งจากผู้ใช้ทั่วไป ดังนั้นการออกแบบในระบบสอบถามข้อมูลสำหรับผู้บริหารฐานข้อมูลจะใช้วิธีการเขียนคำสั่งจากหน้าจอแล้วส่งอาร์กิวเมนต์ที่สำคัญไปเรียกคำสั่งจากโปรแกรมเปลือกของยูนิกซ์ (shell) วิธีนี้จะช่วยให้การทำงานของคำสั่งเป็นไปได้อย่างรวดเร็ว รูปแบบคำสั่งที่เขียนจากหน้าจอคือ

รูปแบบ

sqldba [-**h** host_name] {**create** database_name|**drop**

database_name|**shutdown**|**list** [database_name] [table_name]|**info**}

โดยแต่ละคำมีความหมายดังนี้

sqldba	คือ คำสั่งสำหรับผู้บริหารฐานข้อมูลใช้ติดต่อกับฐานข้อมูล
host_name	หมายถึง ชื่อเครื่องที่ต้องการติดต่อ
database_name	หมายถึง ชื่อฐานข้อมูลที่ต้องการเรียกใช้
table_name	หมายถึง ชื่อตารางภายใต้ฐานข้อมูลนั้น
-h	ใช้บอกกรณีที่ผู้บริหารฐานข้อมูลต้องการติดต่อกับส่วนจัดการฐานข้อมูลที่อยู่ต่างเครื่องกับผู้ใช้
create	คือ คำสั่งให้ส่วนจัดการฐานข้อมูลทำการสร้างฐานข้อมูล
drop	คือ คำสั่งให้ส่วนจัดการฐานข้อมูลทำการลบฐานข้อมูล

shutdown คือ คำสั่งให้ส่วนจัดการฐานข้อมูลทำการปิดฐานข้อมูลและสิ้นสุดการทำงาน

list คือ คำสั่งให้ส่วนจัดการฐานข้อมูลทำการแสดงรายชื่อตามรูปแบบที่ส่ง ซึ่งถ้าไม่มีอะไรตามหลัง list จะหมายถึง การแสดงรายชื่อฐานข้อมูลทั้งหมดที่ส่วนจัดการฐานข้อมูลนั้นดูแลอยู่ ถ้ามี database_name ตามหลัง list หมายถึง การแสดงรายชื่อตารางที่อยู่ภายใต้ database_name แต่ถ้ามี database_name และ table_name จะหมายถึงการแสดงรายชื่อสดมภ์ที่อยู่ภายใต้ table_name และ database_name นั้นๆ

info หมายถึง การดูข้อมูล เช่น ซอคเก็ตที่ใช้ติดต่อ หมายเลขเวอร์ชั่น เป็นต้น

หมายเหตุ [] หมายถึง ในการเขียนคำสั่งจะมีหรือไม่มีข้อความในส่วนนี้ก็ได้

{} หมายถึง ในการเขียนคำสั่งต้องมีข้อความในส่วนนี้กำกับอยู่

| หมายถึง ในการเขียนคำสั่ง ผู้เขียนสามารถทำการเลือกข้อความที่มีเครื่องหมายขั้นอยู่

อักษรตัวหนา หมายถึง คำสั่งจะต้องเขียนตามที่กำหนด

อักษรตัวบาง หมายถึง คำสั่งที่ผู้ใช้สามารถกำหนดเอง

อัลกอริทึมของ sqldba

```
sqldba(argc, argv)
argc is an integer
*argv is a pointer of a character
{
        set home directory
        connect server
        switch (argv)
        case CREATE:
                create new database
```

```
        case DROP:
            drop database
        case SHUTDOWN:
            shutdown server
        case LIST:
            set no. of argument left into argsLeft
            switch (argsLeft)
                case 1:
                    list the available database
                case 2:
                    list the available tables
                case 3:
                    list the attributes and type
        case INFO:
            list information

        close connection
}
```

4.3.2 การออกแบบระบบสำหรับผู้ใช้บริการ

หน้าที่ของผู้ใช้บริการมี ดังนี้

- สร้างและลบตาราง
- จัดการบำรุงรักษาข้อมูลในตารางให้ทันสมัยอยู่เสมอ
- เรียกใช้ข้อมูลในตาราง

เนื่องจากรูปแบบการเรียกใช้ข้อมูลโดยผู้ใช้ข้อมูลมักจะมีความซับซ้อนของการเรียกใช้มากกว่าผู้บริหารฐานข้อมูล และรูปแบบต่างๆนั้นเป็นแบบที่ไม่มีความคงตัว ดังนั้นการ

ออกแบบคำสั่งเพื่อให้ผู้ใช้บริการมีความสะดวกสบายจะต้องมีโปรแกรมบรรณาธิกรณ์ช่วยให้การเขียนคำสั่งมีความคล่องตัวยิ่งขึ้น รูปแบบคำสั่งที่ใช้เรียกระบบสำหรับผู้ใช้บริการคือ

รูปแบบ

sql [**-h** host_name] database_name

โดยแต่ละคำมีความหมายดังนี้

sql คือ คำสั่งสำหรับผู้ใช้บริการใช้ติดต่อกับฐานข้อมูล

host_name หมายถึง ชื่อเครื่องที่ต้องการติดต่อ

database_name หมายถึง ชื่อฐานข้อมูลที่ต้องการเรียกใช้

-h ใช้บอกกรณีที่ผู้ใช้บริการต้องการติดต่อกับส่วนจัดการฐานข้อมูลที่อยู่ต่างเครื่องกับผู้ใช้

อัลกอริทึมของ sql

```
sql()
{
        connect sevrver
        while (not end of file)
                put stdin into inchar
                if (check inchar is '\\') {
                        if (check instring) {
                                put stdin into inchar
                                continue
                        }
                        put stdin into inchar
```

```
        if (check inchar is EOF) {
                continue
        }
        switch (inchar)
        case 'h':
                display help screen
        case 'g':
                send buffer to server
                display return data from server
        case 'e':
                write temporary file from buffer
                edit temporary file
                read temporary file into buffer
        case 'q':
                close connection
        case 'l':
                display buffer
        case 't':
                list the available tables
        case 'f':
                list the attributes and type
}
else
{
        if (check inchar is "") {
```

```
                    if (instring) {
                        set 0 into instring
                    }
                    else
                    {
                        set 1 into instring
                    }
                }
                if (instring) {
                    put inchar into buffer
                    continue
                }
            }
        close connection
    }
```

4.4 การออกแบบระบบจัดการฐานข้อมูล

ระบบจัดการฐานข้อมูล ทำหน้าที่รับคำสั่งจากผู้ใช้บริการ มาแปลความหมายเพื่อนำไปปฏิบัติการกับแฟ้มข้อมูลตามที่ผู้ใช้บริการสั่ง และเมื่อได้ข้อมูลแล้วระบบจะทำการติดต่อกลับไปยังผู้ใช้บริการ โดยส่งผลที่ได้จากการปฏิบัติการแฟ้มข้อมูลกลับสู่ผู้ใช้ต่อไป

เนื่องจากระบบจัดการฐานข้อมูลจะทำงานตลอดเวลา ดังนั้น การออกแบบระบบจะเป็นการประมวลผลส่วนหลัง (Background processing) รูปแบบคำสั่งที่ใช้เรียกระบบจัดการฐานข้อมูลมีดังนี้

รูปแบบ

sqld&

โดยแต่ละคำมีความหมายดังนี้

sqld	คือ คำสั่งสำหรับเรียกระบบจัดการฐานข้อมูล
&	คือ สัญลักษณ์ของยูนิกซ์เพื่อนำกระบวนการไปประมวลผลส่วนหลัง

อัลกอริทึมของ sqld

```
sqld()
{
        set home directory
        open 'lpsock' socket into TCP
        bind to an arbitrary returned address
        place the 'lpsock' in the 'listen state'
        open 'UNIXsock' socket into local communication channel
        bind to an arbitrary returned address
        place the 'UNIXsock' in the 'listen state'
        while (True)
                put 'lpsock' and 'UNIXsock' into rfds set
                if (check number incoming data) {
                        continue
                }
                if (check incoming data from 'lpsock') {
                        put 'lpsock' into sock
                }
```

```
if (check incoming data from 'UNIXsock') {
        put 'UNIXsock' into sock
}
if (sock) {
        open newsock that has connection to the client
        store the connection details
        if (not error) {
                send message to client
                if (check error of received message) {
                        close client connection
                }
                else
                {
                        put newsock into rfds set
                        send message to client
                }
        }
        continue
}
/* this must be command */
set 3 into comsock
while (comsock < 100)
        if (check incoming data from comsock) {
                put comsock into rfds set
```

```
if (check error of received message) {
        put 'QUIT' into command
}
else
{
        put first byte of received message
                into command
}
switch (command)
        case INIT_DB:
                open database
        case QUERY:
                process query command
        case DB_LIST:
                list the available database
        case TABLE_LIST:
                list the available tables
        case FIELD_LIST:
                list the attributes and type
        case QUIT:
                close client connection
        case CREATE_DB:
                create database
        case DROP_DB:
                drop database
```

```
                                    case SHUTDOWN:
                                        close client connection
                                        close 'Ipsock' and 'UNIXsock'
                                }
                                increase comsock
        }
```

4.5 การออกแบบระบบการแปลงผันข้อมูล-dBASE

เมื่อพิจารณารูปแบบของแฟ้มข้อมูล dBASE และแฟ้มข้อมูลบนยูนิกซ์ ประกอบกับคุณสมบัติของฐานข้อมูลสมบูรณ์ เช่น Oracle และ Sybase แล้วนั้น จะเห็นได้ว่า การนำแฟ้มข้อมูลจาก dBASE มาใช้บนยูนิกซ์ จะต้องมีการแปลงให้ข้อมูลอยู่ในสภาพที่สามารถใช้งานได้ดีบนระบบการใช้ข้อมูลร่วมกัน ดังนั้นระบบการแปลงผันข้อมูล-dBASE จะมีหน้าที่แปลงส่วนหัวของแฟ้มข้อมูล dBASE ให้มาอยู่ในรูปการสร้างตาราง และแปลงส่วนข้อมูล dBASE ให้มาอยู่ในรูปการเพิ่มแนวนอนในตาราง

เนื่องจากการระบบการแปลงผันข้อมูล dBASE จะต้องทำการประมวลผลข้อมูลทั้งแฟ้มข้อมูลและรูปแบบคำสั่งมีเพียงรูปแบบเดียว ดังนั้น การออกแบบหน้าจอจะเป็นลักษณะเขียนคำสั่งจากโปรแกรมเปลือกของยูนิกซ์ (Shell) โดยใช้อาร์กิวเมนต์เป็นตัวแยกสิ่งที่ผู้ใช้ต้องการให้ทำงาน รูปแบบคำสั่งที่ใช้เรียกระบบการแปลงผันข้อมูล dBASE คือ

รูปแบบ

db2sql [-ul-l] [-**h** host_name] [-**p** primary_key]

-**d** database_name -**t** table_name [-**c**] [-**v**] dBase_file

โดยแต่ละคำมีความหมายดังนี้

db2sql	คือคำสั่งสำหรับระบบการแปลงผันข้อมูลจาก dBASE เป็นแฟ้มข้อมูลที่สามารถเรียกใช้ได้จากระบบสอบถามข้อมูล
host_name	หมายถึง ชื่อเครื่องที่ต้องการติดต่อ
primary_key	หมายถึง ชื่อสดมภ์ที่เป็นกุญแจหลัก
database_name	หมายถึง ชื่อฐานข้อมูลที่ต้องการเรียกใช้

table_name	หมายถึง ชื่อตารางภายใต้ฐานข้อมูลนั้น
dBase_file	หมายถึง ชื่อแฟ้มข้อมูล dBASE ที่ต้องการแปลงผัน
-u	ใช้บอกกรณีที่ผู้ใช้ต้องการแปลงข้อมูลเป็นตัวพิมพ์ตัวใหญ่ (Uppercase)
-l	ใช้บอกกรณีที่ผู้ใช้ต้องการแปลงข้อมูลเป็นตัวพิมพ์ตัวเล็ก (Lowercase)
-h	ใช้บอกกรณีที่ผู้ใช้ต้องการติดต่อกับส่วนจัดการฐานข้อมูลที่อยู่ต่างเครื่องกับผู้ใช้
-p	ใช้บอกกรณีที่ผู้ใช้ต้องการมีกุญแจหลัก
-d	ใช้บอกกรณีที่ผู้ใช้ต้องการติดต่อกับฐานข้อมูล
-t	ใช้บอกกรณีที่ผู้ใช้ต้องการติดต่อกับตาราง
-c	ใช้บอกกรณีที่ผู้ใช้ต้องการสร้างตารางใหม่
-v	ใช้บอกกรณีที่ผู้ใช้ต้องการข้อความรายงานการทำงาน

อัลกอริทึมของ db2sql

```
db2sql(argc, argv)
argc is integer
*argv is pointer of a character
{
        set arguments into variables
        open dbase file
        connect  server
        read the header part of dbase file into header
        if (not create) {
                if (check the existence of table) {
                        return
                }
```

```
        }
        else
        {
                drop table
                convert from header to create command
                create table
        }
        read the data part of dbase file into buffer
        while (not last record)
                convert from buffer to insert command
                send query to server
                read the data part of dbase file into buffer
        close dbase file
        close connection
}
```