# ภาคผนวก ก

## แผนภาพวากยสัมพันธ์

ในการอธิบายวากยสัมพันธ์ของตัวแปลภาษาสอบถามฐานข้อมูลพีชคณิตสัมพันธ์สำหรับ MiniRDBMS ได้ใช้แผนภาพวากยสัมพันธ์ (Syntax Diagram) ในการอธิบาย แผนภาพแสดงด้วยรูปภาพที่มีสัญลักษณ์ที่ถูกแทนที่ได้หรือสัญลักษณ์ที่ถูกแทนที่ไม่ได้กำกับไว้ แต่ละรูปภาพจะมีลูกศรเข้าทางเดียว และออกทางเดียวเสมอ มีกฎเกณฑ์ ดังนี้

1) สิ่งใด ๆ ที่ปรากฏในรูป ◯ หรือ ▭ จะต้องเขียนสิ่งนั้นในแถวคำสั่งเพื่อสอบถามเหมือนที่ปรากฏในแผนภาพ โดยที่ตัวอักษรตัวใหญ่หรือตัวเล็กไม่แตกต่างกัน

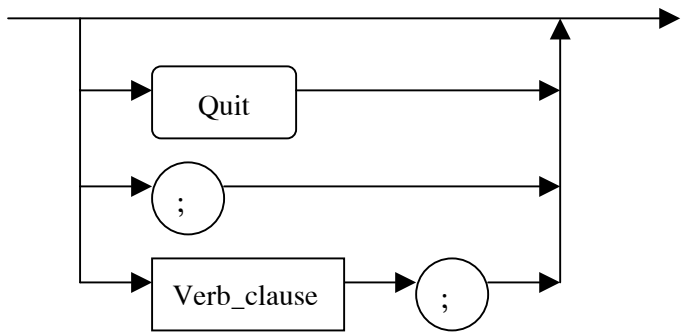
2) ทุกข้อความที่ปรากฏในรูป ▭ เป็นสัญลักษณ์ที่ถูกแทนที่ได้

3) Table_name ในแผนภาพคือชื่อตารางข้อมูล

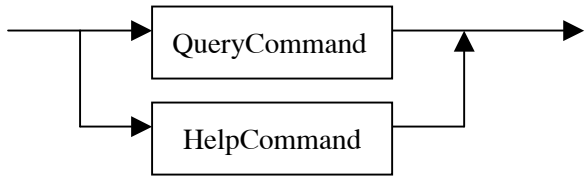
4) Attribute_name ในแผนภาพคือชื่อแอตทริบิว

5) Value ในแผนภาพคือค่าคงที่

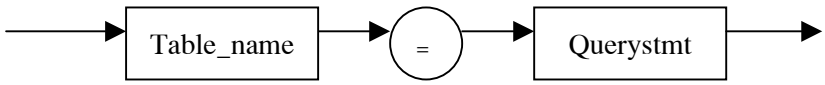
Query



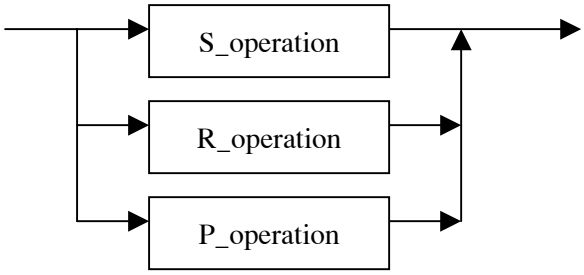

Verb_clause



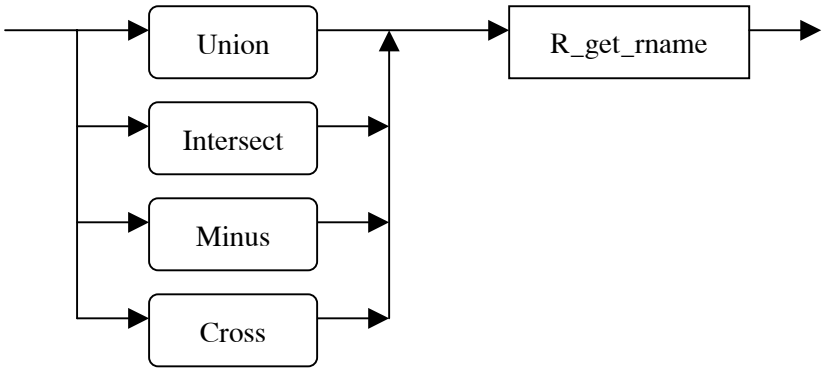

QueryCommand



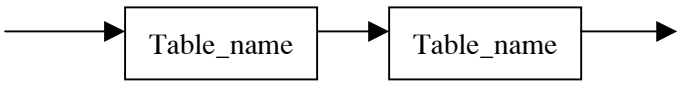
Querystmt



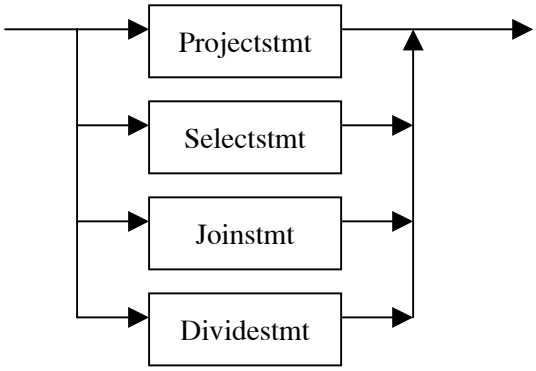

S_operation



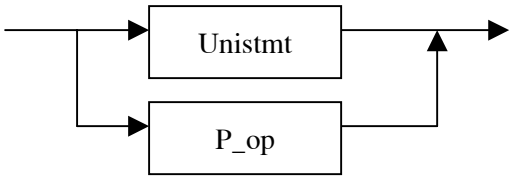

R_get_rname



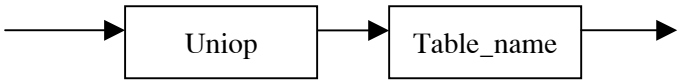

R_operation



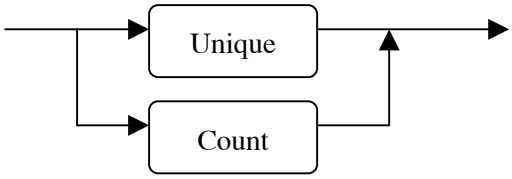
P_operation



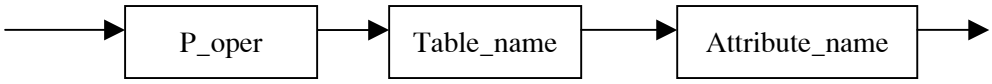

Unistmt



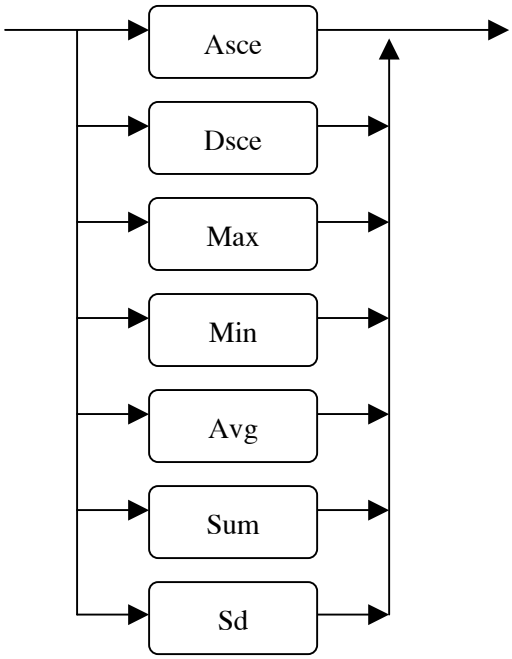

Uniop




P_op

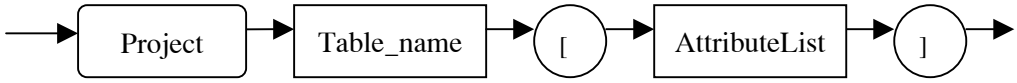



P_oper

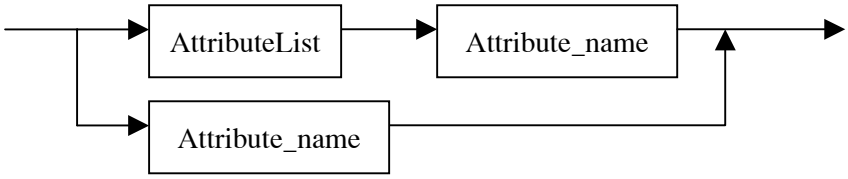


P_parameter

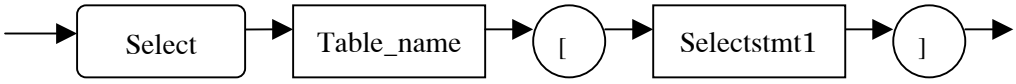

Projectstmt



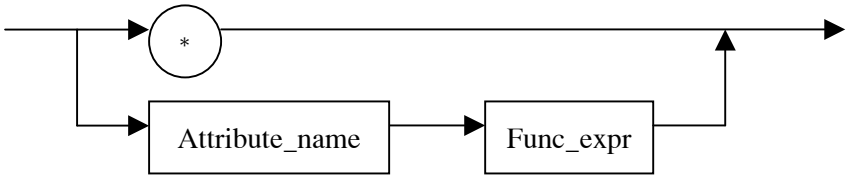

AttributeList



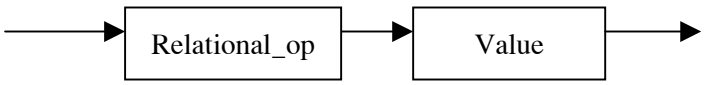

Selectstmt



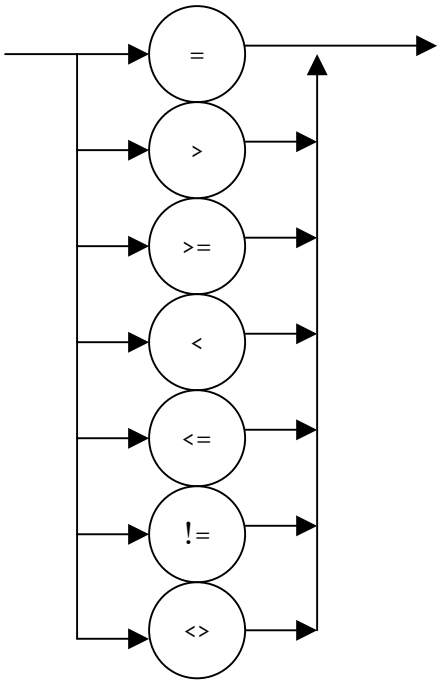

Selectstmt1



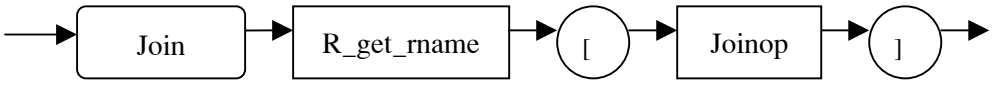

Func_expr



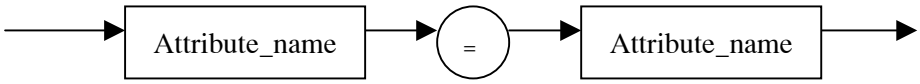
Relational_op



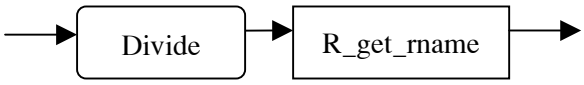

Joinstmt



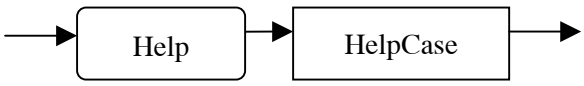

Joinop



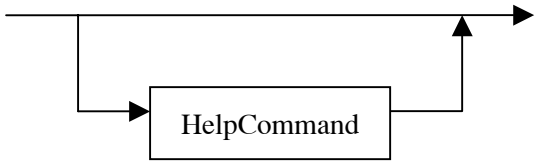

Dividestmt



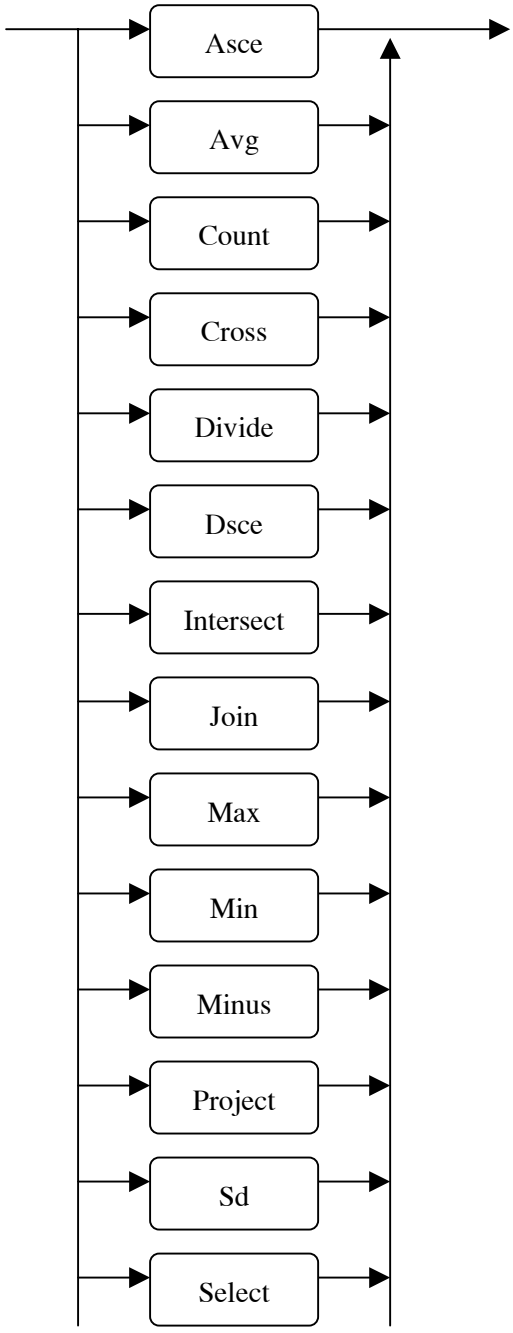

HelpCommand

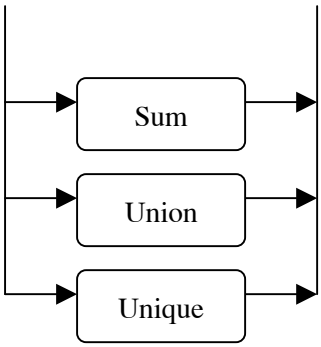


HelpCase




HelpCommand




Sum
Union
Unique

# ภาคผนวก ข

## ไวยากรณ์ของคำสั่งปฏิบัติการ

เมื่อผู้ใช้ต้องการใช้งานระบบจะต้องพิมพ์คำสั่งปฏิบัติการ ซึ่งผู้ใช้จะต้องพิมพ์ให้ถูกต้องตามไวยากรณ์และรูปแบบคำสั่งที่ได้ออกแบบไว้ ซึ่งไวยากรณ์ของวิทยานิพนธ์นี้จะนำไปรวมกับไวยากรณ์ของวิทยานิพนธ์เดิม ทำให้ผู้ใช้สามารถใช้คำสั่งที่เกี่ยวกับการจัดการข้อมูลในฐานข้อมูลและสอบถามข้อมูลในฐานข้อมูลได้ในขณะเดียวกัน ไวยากรณ์ของคำสั่งปฏิบัติการทั้งหมดในงานวิทยานิพนธ์นี้มีดังนี้

```
%{
#include "lex.h"
#define Lex CurrentLex
#define COL (unsigned int) (Lex->tok_start - Lex->sqlstr)+1
extern void yyerror(const char*);

%}
%token  TK_SEMICOLON
%token  TK_EQ
%token  TK_GE
%token  TK_GT
%token  TK_LE
%token  TK_LT
%token  TK_NE
%token  TK_RBCK
%token  TK_LBCK
%token  TK_COUNT
%token TK_CROSS
%token TK_DIVIDE
%token TK_DSCE
%token TK_HELP
%token TK_INTERSECT
%token  TK_JOIN
```

```
%token TK_MAX
%token  TK_MIN
%token  TK_MINUS
%token TK_PROJECT
%token TK_SELECT
%token TK_SUM
%token TK_UNION
%token TK_UNIQUE
%token TK_VALUES
%token TK_ASCE
%token TK_AVG
%token  TK_SD
%token TK_DATE
%token TK_IDENT
%token  TK_STRING
%token TK_INTEGER
%token TK_REAL
%token  TK_QUIT

%type  <string> TK_IDENT TK_STRING TK_INTEGER TK_REAL dbname
                relname attname
%type  <num>  relational_op func_expr function_expr type value
              op_delete_on
%type  <NONE>          '(' ')' ','

%%

query:  /* empty */ { YYABORT; }
        | TK_QUIT    { Lex->SqlCommand = SQLCOM_QUIT;}
        | TK_SEMICOLON  { YYABORT; }
        | verb_clause end_semi {}

end_semi:/* empty */
```

```
        {
            PrintErrPosition(COL);
          ShowErrMsg(ERR_SEMICOLON);
          YYABORT;
       }
        | TK_SEMICOLON {}

verb_clause : querycommand
            | helpcommand;

dbname:     TK_IDENT
           {
            if (($1.length) > NAMELEN )
            {
              PrintErrPosition(COL);
              ShowErrMsg(ERR_NAMELEN,NAMELEN);
              YYABORT;
            }
            if (!CheckIdentName($1.str,COL))
                YYABORT;
            StrUpr($1.str);
            $$ = $1;
           }

relname:   TK_IDENT
          {
            if (($1.length) >= NAMELEN )
            {
              PrintErrPosition(COL);
              ShowErrMsg(ERR_NAMELEN,NAMELEN);
              YYABORT;
            }
            if (!CheckIdentName($1.str,COL))
```

```
            YYABORT;
        StrUpr($1.str);
        $$ = $1;
    }

attname:  TK_IDENT
    {
        if (($1.length) >= NAMELEN )
        {
            PrintErrPosition(COL);
            ShowErrMsg(ERR_NAMELEN,NAMELEN);
            YYABORT;
        }
        if (!CheckIdentName($1.str,COL))
            YYABORT;
        StrUpr($1.str);
        $$ = $1;
    }
        {
        if (!(SetStrNameNew($1)))
          YYABORT;
        }

          TK_EQ  querystmt;

querycommand : relname
        {
        if (!(SetStrNameNew($1)))
          YYABORT;
        }

          TK_EQ  querystmt;
```

```
querystmt : s_operation
          | r_operation
          | p_operation;

s_operation : TK_INTERSECT
              {
                Lex->SqlCommand    = SQLCOM_INTERSECT;
              }
                r_get_rname

            | TK_UNION
              {
                Lex->SqlCommand    = SQLCOM_UNION;
              }
                r_get_rname

            | TK_MINUS
                {
                 Lex->SqlCommand    = SQLCOM_MINUS;
              }
                r_get_rname

             | TK_CROSS
              {
                Lex->SqlCommand    = SQLCOM_CROSS;
              }
                r_get_rname ;

r_get_rname : relname
                {
                  if (!(SetStrName($1)))
                YYABORT;
                }
```

```
                relname
                  {
                      if (!(SetStrNameLast($3)))
                   YYABORT;
                  }

r_operation : projectstmt
            | selectstmt
            | joinstmt
            | dividestmt;

projectstmt : TK_PROJECT
                  {
                  ListEmpty(&Lex->ListAttName);
                  ListEmpty(&Lex->ListAtt);
                  Lex->SqlCommand    = SQLCOM_PROJECT;
                }
                  relname
                  {
                  if (!(SetStrName($3)))
                  YYABORT;
                  }
                  TK_LBCK attlist TK_RBCK;

attlist  : attlist attname
                  {
                  if (!(AddColToList((STRING)$2,0)))
                   YYABORT;
                   }
           | attname
                  {
                   if (!(AddColToList((STRING)$1,0)))
                          YYABORT;
```

```
        }

selectstmt : TK_SELECT
            {
               ListEmpty(&Lex->ListAttName);
             ListEmpty(&Lex->ListExpr);
               Lex->SqlCommand    = SQLCOM_SELECT_AL;
             }
             relname
             {
               if (!(SetStrName($3)))
              YYABORT;
             }
             TK_LBCK selectstmt1 TK_RBCK;

selectstmt1 : '*'
          | attname
            {
            Lex->Col = COL;
          }
          func_expr
          {
              if (!AddExprToList($1,(enum FUNCID)$3,
                                 &(Lex->ListItem)))
                        YYABORT;
            }

func_expr : relational_op value1
joinstmt : TK_JOIN
             {
              ListEmpty(&Lex->ListAttName);
               Lex->SqlCommand    = SQLCOM_JOIN;
             }
```

```
            r_get_rname TK_LBCK joinop TK_RBCK;

joinop : attname
            {
           if(!(AddColToList((STRING)$1,0)))
                    YYABORT;
          }
        joinop2 ;

joinop2 : TK_EQ   attname
            {
             if(!(AddColToList((STRING)$2,0)))
                    YYABORT;
          }

dividestmt : TK_DIVIDE
            {
             Lex->SqlCommand    = SQLCOM_DIVIDE;
            }
          r_get_rname;
p_operation : unistmt
           | p_op;

unistmt : uniop relname
            {
              if (!(SetStrName($2)))
             YYABORT;
            }
uniop : TK_UNIQUE
            {
             Lex->SqlCommand    = SQLCOM_UNIQUE;
            }
      | TK_COUNT
```

```
            {
                Lex->SqlCommand     = SQLCOM_COUNT;
              }

   p_op : p_oper  relname
              {
                if (!(SetStrName($2)))
                 YYABORT;
              }

         p_parameter;

p_oper : TK_ASCE
            {
                ListEmpty(&Lex->ListAttName);
               Lex->SqlCommand     = SQLCOM_ASCE;
              }

        | TK_DSCE
            {
                ListEmpty(&Lex->ListAttName);
                Lex->SqlCommand     = SQLCOM_DSCE;
              }
         | TK_MAX
            {
                ListEmpty(&Lex->ListAttName);
                Lex->SqlCommand     = SQLCOM_MAX;
              }

      | TK_MIN
            {
                ListEmpty(&Lex->ListAttName);
                Lex->SqlCommand     = SQLCOM_MIN;
```

```
            }
        | TK_SUM
            {
                ListEmpty(&Lex->ListAttName);
                Lex->SqlCommand     = SQLCOM_SUM;
            }
        | TK_AVG
            {
                ListEmpty(&Lex->ListAttName);
                Lex->SqlCommand     = SQLCOM_AVG;
            }
        | TK_SD
            {
                ListEmpty(&Lex->ListAttName);
                Lex->SqlCommand     = SQLCOM_SD;
            }

p_parameter : TK_LBCK attname
            {
              if(!(AddColToList((STRING)$2,0)))
                    YYABORT;
              }
          TK_RBCK;


helpcommand : TK_HELP helpcase;

helpcase : { Lex->SqlCommand     = SQLCOM_HELP;}
        | helpcommand ;

helpcommand : TK_ASCE { Lex->SqlCommand     = SQLCOM_HELPASCE;}
            | TK_AVG  { Lex->SqlCommand     = SQLCOM_HELPAVG;}
            | TK_COUNT {Lex->SqlCommand     = SQLCOM_HELPCOUNT;}
```

```
        | TK_CROSS {Lex->SqlCommand    = SQLCOM_HELPCROSS;}
        | TK_DIVIDE {Lex->SqlCommand    = SQLCOM_HELPDIVIDE;}
        | TK_DSCE {Lex->SqlCommand    = SQLCOM_HELPDSCE;}
        | TK_INTERSECT {Lex->SqlCommand    = SQLCOM_HELPINTERSECT;}
        | TK_JOIN {Lex->SqlCommand    = SQLCOM_HELPJOIN;}
        | TK_MAX {Lex->SqlCommand    = SQLCOM_HELPMAX;}
        | TK_MIN {Lex->SqlCommand    = SQLCOM_HELPMIN;}
        | TK_MINUS {Lex->SqlCommand    = SQLCOM_HELPMINUS;}
        | TK_PROJECT {Lex->SqlCommand    = SQLCOM_HELPPROJECT;}
        | TK_SD {Lex->SqlCommand    = SQLCOM_HELPSD;}
        | TK_SELECT {Lex->SqlCommand    = SQLCOM_HELPSELECT;}
        | TK_SUM {Lex->SqlCommand    = SQLCOM_HELPSUM;}
        | TK_UNION {Lex->SqlCommand    = SQLCOM_HELPUNION;}
        | TK_UNIQUE {Lex->SqlCommand    = SQLCOM_HELPUNIQUE;}

%%
void PrintErrPosition(short col)
{
  char format[10],str1[255], *mark="^";
  int i  = strlen(CurrentLex->sqlstr),j=0;
  char *str=(char*)str1;

  if (col > 80)
  {
     col = col % 80;
     j  = i/80;
     i  = i - j;
     j  = (j-1)*80;
  }

  if (i > 80) i = 80;
  *str++='\n';
  StrMovN(str,CurrentLex->sqlstr+j,i);
```

```
    (void)PrintMessage(str1);
    str=(char*)str1;
    *str++='\n';
    for(i=0 ;i < col-1 ; i++)
        *str++ = ' ';
    *str++ = '^';
    *str++ ='\n';
    *str++ ='\0';
    (void)PrintMessage(str1);
}

void yyerror(const char *s)
{
  (void) PrintErrPosition(COL);
  ShowErrMsg(ERR_SYNTAX,CurrentLex->tok_start);
}

BOOL CheckIdentName(char *name,int col)
{
  /* Must start with character */
  if (!isalpha(*name))
     goto err;
  /* Next character is number or character or _ */
  for (name++ ; *name ; name++)
    if (!ISVAR(*name))
        goto err;
  return (1);
err:
  PrintErrPosition(col);
  ShowErrMsg(ERR_NAME);
  return(0);
}
```

```
short SetStrName(STRING strname)
{
   if ((strname.length) >= NAMELEN )
   {
      PrintErrPosition(COL);
      ShowErrMsg(ERR_NAMELEN,NAMELEN);
      return(0);
   }
   if (Lex->StrName.Str)
   {
       free(Lex->StrName.Str);
       Lex->StrName.Str=0;
   }
   Lex->StrName.Str=(char*)StrDupN(strname.str,strname.length);
   Lex->StrName.Len = strname.length;
   Lex->StrName.Col = COL;
   free(strname.str);
   return(1);
}

short SetStrNameNew(STRING strnamenew)
{
   if ((strnamenew.length) >= NAMELEN )
   {
      PrintErrPosition(COL);
      ShowErrMsg(ERR_NAMELEN,NAMELEN);
      return(0);
   }
   if (Lex->StrNew.Str)
   {
       free(Lex->StrNew.Str);
       Lex->StrNew.Str=0;
   }
```

```
    Lex->StrNew.Str=(char*)StrDupN(strnamenew.str,strnamenew.length);
    Lex->StrNew.Len = strnamenew.length;
    Lex->StrNew.Col = COL;
    free(strnamenew.str);
    return(1);
}
short SetStrNameLast(STRING strnamelast)
{
    if ((strnamelast.length) >= NAMELEN )
    {
        PrintErrPosition(COL);
        ShowErrMsg(ERR_NAMELEN,NAMELEN);
        return(0);
    }
     if (Lex->StrName2.Str)
    {
         free(Lex->StrName2.Str);
         Lex->StrNew.Str=0;
    }
    Lex->StrName2.Str=(char*)StrDupN(strnamelast.str,strnamelast.length);
    Lex->StrName2.Len = strnamelast.length;
    Lex->StrName2.Col = COL;
    free(strnamelast.str);
    return(1);
}
```

# ภาคผนวก ค

## ตารางรหัสโทเคนของคำสั่งปฏิบัติการในระบบ

เมื่อผู้ใช้มีการพิมพ์แถวคำสั่ง ระบบจะเข้าสู่ขั้นตอนการแยกและวิเคราะห์ศัพท์เพื่อแยกข้อความสั่งนั้นเป็นโทเคน จากนั้นจะเข้าสู่ขั้นตอนการวิเคราะห์วากยสัมพันธ์เพื่อตรวจสอบว่าแต่ละโทเคนที่รับมานั้นถูกต้องตามหลักไวยากรณ์หรือไม่ โดยจะต้องมีการเปรียบเทียบโทเคนกับตารางรหัสโทเคนเพื่อหารหัสของโทเคนนั้น ๆ

โปรแกรมที่สร้างตารางรหัสโทเคนของคำสั่งปฏิบัติการในระบบ ได้แก่ โปรแกรม lex.h ซึ่งภายในโปรแกรมจะมีการกำหนดรหัสของโทเคนของวิทยานิพนธ์เดิมไว้ ผู้วิจัยจึงได้ทำการเพิ่มเติมสัญลักษณ์และรหัสโทเคนของวิทยานิพนธ์ชิ้นนี้ลงไป โดยเรียงลำดับตามรหัส ASCII ดังนี้

```
#include ″data_st.h″
#include ″y.tab.h″
#define CurrentLex (&CurrLex)
#define TOKEN(A) A
enum LEXSTATE { STATE_START, STATE_CHAR, STATE_IDENT,
                STATE_REAL,
                STATE_CMP_OP,
                STATE_STRING,
                STATE_END,
                STATE_NUMBER,
                STATE_POINT,
                STATE_SEMICOLON,
                STATE_EOL,
                STATE_RBCK,
                STATE_LBCK
 };
typedef union {
  int num;
  STRING string;
} YYSTYPE;
```

```
typedef struct lex_st {
  unsigned int   yytoklen;  /* Simulate lex */
  YYSTYPE yylval;
  short Col;
  unsigned char *sqlstr, *ptr, *tok_start,*end_of_query;
  enum SQLCOMMAND SqlCommand;
  STRNAME StrName,StrNew,StrName2;
  int Length,Decimal;
  ITEMNODE DefaValue;
  char Domain;
  unsigned long Flag;
  unsigned char PKey;
  unsigned char SKey;
  LIST ListAtt;      /* user for created attribute node */
  LIST ListPKey;     /* list primary key */
  LIST ListSKey;     /* list secondary key */
  LIST ListFKey;     /* list foreigned key */
  LIST ListRef;      /* list foreign key */
  LIST ListExpr;     /* use for where clause */
  LIST ListAttName; /* list column name insertinto,delete,select*/
  LIST ListItem;     /* list values  for insertinto */
  ITEMUPD *ListUpd;      /* list values  for upadate*/
}LEX;

typedef struct st_symbol {
  char  *name;
  unsigned int  tok;
  unsigned int length;
} SYMBOL;

static SYMBOL symbols[] = {
  { "!=",                TOKEN(TK_NE),0},
  { "<",                 TOKEN(TK_LT),0},
```

```
{ "<=",          TOKEN(TK_LE),0},
{ "<>",          TOKEN(TK_NE),0},
{ "=",           TOKEN(TK_EQ),0},
{ ">",           TOKEN(TK_GT),0},
{ ">=",          TOKEN(TK_GE),0},
{ "AND",         TOKEN(TK_AND),0},
{ "ASCE",        TOKEN(TK_ASCE),0},
{ "AVG",         TOKEN(TK_AVG),0},
{ "BETWEEN",     TOKEN(TK_BETWEEN),0},
{ "CASCADE",     TOKEN(TK_CASCADE),0},
{ "CHAR",        TOKEN(TK_CHAR),0},
{ "CHARACTER",   TOKEN(TK_CHAR),0},
{ "CLOSE",       TOKEN(TK_CLOSE),0},
{ "COUNT",       TOKEN(TK_COUNT),0},
{ "CREATE",      TOKEN(TK_CREATE),0},
{ "CROSS",       TOKEN(TK_CROSS),0},
{ "DATABASE",    TOKEN(TK_DATABASE),0},
{ "DATABASES",   TOKEN(TK_DATABASES),0},
{ "DATE",        TOKEN(TK_DATE),0},
{ "DEFAULT",     TOKEN(TK_DEFAULT),0},
{ "DELETE",      TOKEN(TK_DELETE),0},
{ "DESC",        TOKEN(TK_DESC),0},
{ "DESCRIBE",    TOKEN(TK_DESCRIBE),0},
{ "DIVIDE",      TOKEN(TK_DIVIDE),0},
{ "DROP",        TOKEN(TK_DROP),0},
{ "DSCE",        TOKEN(TK_DSCE),0},
{ "FOREIGN",     TOKEN(TK_FOREIGN),0},
{ "FROM",        TOKEN(TK_FROM),0},
{ "HELP",        TOKEN(TK_HELP),0},
{ "IN",          TOKEN(TK_IN),0},
{ "INSERT",      TOKEN(TK_INSERT),0},
{ "INTERSECT",   TOKEN(TK_INTERSECT),0},
{ "INTO",        TOKEN(TK_INTO),0},
```

```
{ "JOIN",          TOKEN(TK_JOIN),0},
{ "KEY",           TOKEN(TK_KEY),0},
{ "MAX",           TOKEN(TK_MAX),0},
{ "MIN",           TOKEN(TK_MIN),0},
{ "MINUS",         TOKEN(TK_MINUS),0},
{ "NOT",           TOKEN(TK_NOT),0},
{ "NULL",          TOKEN(TK_NULL),0},
{ "NUM",           TOKEN(TK_NUM),0},
{ "ON",            TOKEN(TK_ON),0},
{ "PRIMARY",       TOKEN(TK_PRIMARY),0},
{ "PROJECT",       TOKEN(TK_PROJECT),0},
{ "QUIT",          TOKEN(TK_QUIT),0},
{ "REFERENCES",    TOKEN(TK_REFERENCE),0},
{ "SD",            TOKEN(TK_SD),0},
{ "SECONDARY",     TOKEN(TK_SECONDARY),0},
{ "SELECT",        TOKEN(TK_SELECT),0},
{ "SET",           TOKEN(TK_SET),0},
{ "SHOW",          TOKEN(TK_SHOW),0},
{ "SUM",           TOKEN(TK_SUM),0},
{ "TABLE",         TOKEN(TK_TABLE),0},
{ "TABLES",        TOKEN(TK_TABLES),0},
{ "UNION",         TOKEN(TK_UNION),0},
{ "UNIQUE",        TOKEN(TK_UNIQUE),0},
{ "UPDATE",        TOKEN(TK_UPDATE),0},
{ "USE",           TOKEN(TK_USE),0},
{ "VALUES",        TOKEN(TK_VALUES),0},
{ "WHERE",         TOKEN(TK_WHERE),0},
{ "[",             TOKEN(TK_LBCK),0},
{ "]",             TOKEN(TK_RBCK),0}
};

LEX CurrLex;
```

# ภาคผนวก ง

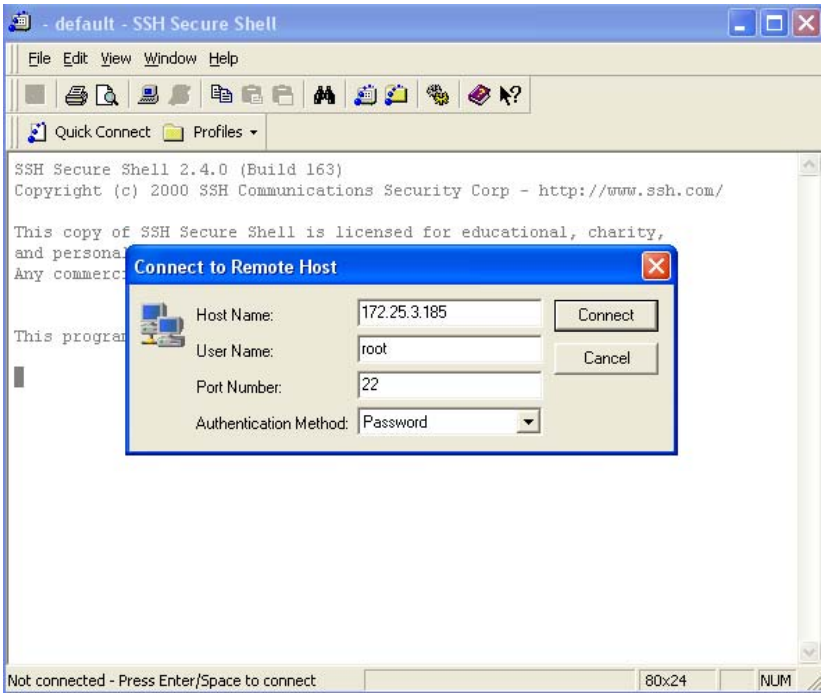
# คู่มือการใช้งานระบบ

ตัวแปลภาษาสอบถามฐานข้อมูลพีชคณิตสัมพันธ์ที่ทำงานอยู่ในรูปแบบของแถวคำสั่ง เนื่องจากงานวิจัยเดิมคือ MiniRDBMS เป็นโปรแกรมที่ทำงานในระบบปฏิบัติการ Linux ดังนั้นในงานวิทยานิพนธ์ชิ้นนี้จึงเป็นรูปแบบแถวคำสั่งในระบบปฏิบัติการ Linux และผู้ใช้สามารถใช้งานระบบที่พัฒนาขึ้นโดยการพิมพ์คำสั่งปฏิบัติการที่ได้กำหนดไว้ให้ถูกต้องตามไวยากรณ์ของคำสั่งนั้น ๆ

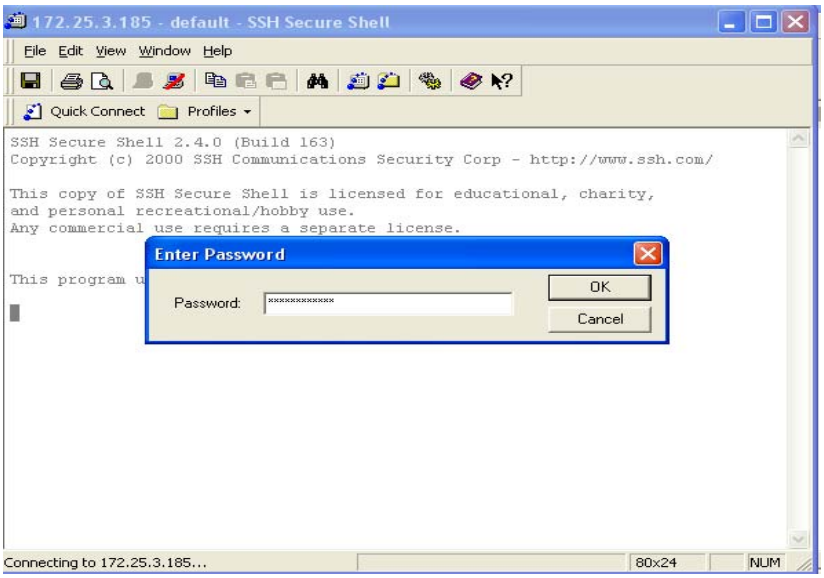
## 1. การเริ่มเข้าสู่การทำงานของระบบ

การใช้งานระบบจะต้องเข้าสู่การทำงานภายใต้ระบบปฏิบัติการ Linux ก่อน โดยจะเข้าสู่การทำงานของระบบที่เครื่องที่มีระบบปฏิบัติการ Linux โดยตรง หรือจะใช้วิธีการเชื่อมต่อผ่านเครื่องคอมพิวเตอร์อื่น ๆ โดยผ่านเครือข่ายด้วยโปรแกรม SSH Secure Shell เพื่อติดต่อไปยังเครื่องที่ทำงานภายใต้ระบบปฏิบัติการ Linux สำหรับเครื่องที่ใช้ในการพัฒนาระบบได้กำหนดหมายเลข IP (Internet Protocol) ไว้เป็น 172.25.3.185

เมื่อจะเข้าสู่การทำงานภายใต้ระบบปฏิบัติการ Linux ผู้ใช้จะต้องใส่ Login และ Password




ภาพประกอบ ง-1 การใส่ Login เพื่อติดต่อกับระบบ




ภาพประกอบ ง-2 การใส่ Password เพื่อติดต่อกับระบบ

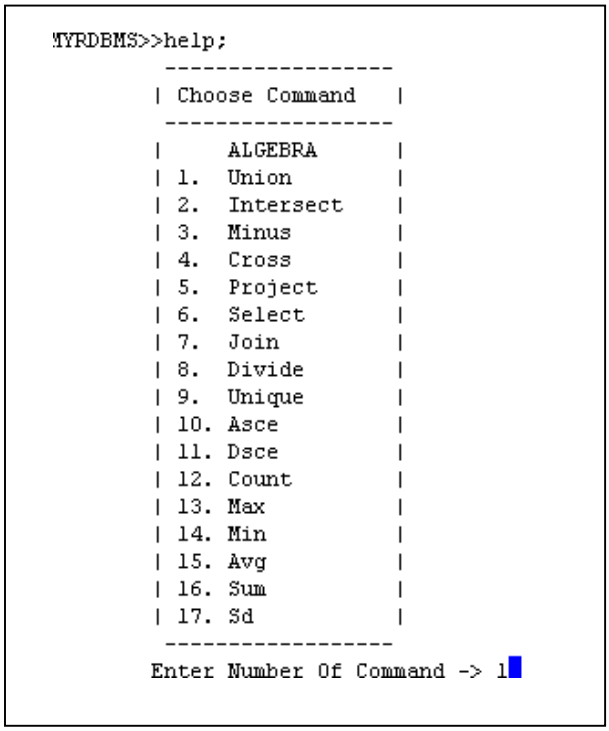
จากนั้นพิมพ์คำสั่ง “./myrdbms” เพื่อเริ่มต้นใช้งานระบบ ระบบจะเริ่มทำงาน โดยแสดงข้อความ “myrdbms>>” ซึ่งเป็น Prompt ของระบบ

```
[root@localhost myalgebra]# ./myrdbms
MYRDBMS>>
```

ภาพประกอบ ง-3 เริ่มใช้งานระบบ

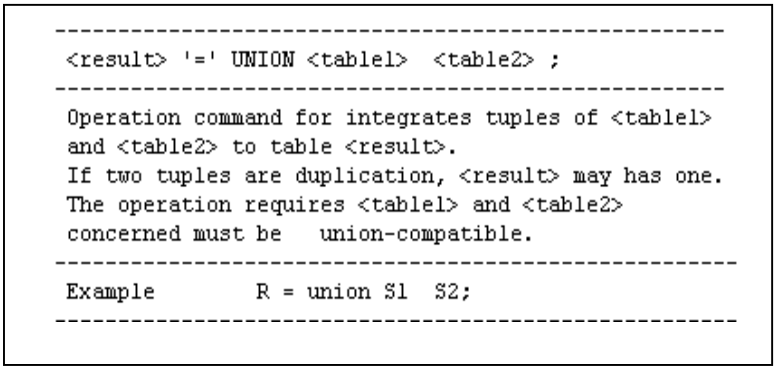
จากนั้นผู้ใช้จะต้องพิมพ์คำสั่งปฏิบัติการที่ต้องการทำงาน ในกรณีที่ผู้ใช้ยังไม่ทราบไวยากรณ์ของคำสั่งปฏิบัติการแต่ละคำสั่ง ในระบบจะมีตัวช่วยให้ผู้ใช้สามารถดูรูปแบบไวยากรณ์ของคำสั่งนั้น ๆ ได้ โดยใช้คำสั่ง 2 รูปแบบ คือ

1. พิมพ์คำสั่ง “Help” จากนั้นระบบจะขึ้นเมนูให้เลือกคำสั่งปฏิบัติการ




ภาพประกอบ ง-4 เมนูของคำสั่ง Help

เมื่อกดเลือกคำสั่งใดคำสั่งหนึ่งแล้ว ระบบจะแสดงไวยากรณ์ของคำสั่งนั้นขึ้นมา




ภาพประกอบ ง-5 ไวยากรณ์คำสั่งปฏิบัติการ Union

2. พิมพ์คำสั่ง “Help” และตามด้วยคำสั่งปฏิบัติการที่ต้องการทราบ

```
help intersect;
```

จากนั้นระบบจะแสดงไวยากรณ์ของคำสั่งนั้น ๆ

```
MYRDBMS>>help intersect;
        ------------------------------------------------------
         <result>  '='   INTERSECT <table1>  <table2> ;
        ------------------------------------------------------
         Operation command for chooses tuples
         which have in both <table1> and <table2>
         to table <result> . The operation requires <table1>
         and <table2> concerned must be union-compatible.
        ------------------------------------------------------
         Example        R = intersect   S1  S2;
        ------------------------------------------------------
MYRDBMS>>
```

ภาพประกอบ ง-6 ไวยากรณ์คำสั่งปฏิบัติการ Intersect

## 2. ตัวอย่างการใช้งานระบบ

การใช้งานระบบผู้ใช้จะต้องพิมพ์คำสั่งปฏิบัติการตามไวยากรณ์ของแต่ละคำสั่ง โดยที่ทุกคำสั่งจะต้องลงท้ายด้วยเครื่องหมาย ‘ ; ’ เสมอ เมื่อผู้ใช้พิมพ์คำสั่งเรียบร้อยแล้วให้กดปุ่ม Enter ระบบจะแสดงผลลัพธ์การทำงานของคำสั่งปฏิบัติการที่ผู้ใช้พิมพ์เข้าไป แต่ถ้าผู้ใช้พิมพ์ไม่ถูกต้องจะแสดงข้อความแจ้งให้ผู้ใช้ทราบ

สมมติมีตารางข้อมูลต่อไปนี้

ตาราง DEPT1

| DEPT_ID | DEPT_NAME |
|---|---|
| 1 | Compsc |
| 2 | Math |
| 3 | Physics |
| 4 | Chem |

ตาราง DEPT2

| DEPT_ID | DEPT_NAME |
|---|---|
| 1 | compsc |
| 1 | compsc |
| 5 | bio |
| 6 | biochem |
| 4 | chem |

ตาราง STUDENT

| STD_ID | STD_NAME | DATE | SEX | HIGH | WEIGHT | DEPT_ID |
|---|---|---|---|---|---|---|
| 1 | Somsri | 2/11/2523 | F | 165 | 55 | 1 |
| 2 | Somchai | 21/2/2522 | M | 180 | 70 | 1 |
| 3 | Sompong | 15/8/2526 | F | 155 | 42 | 2 |
| 4 | Somwang | 6/5/2522 | M | 172 | 68 | 4 |

ตาราง SUBJ_ID

| SUBJ_ID |
|---|
| 1 |
| 2 |

ตาราง STD_SUBJ

| STD_ID | SUBJ_ID |
|---|---|
| 1 | 1 |
| 1 | 3 |
| 3 | 1 |
| 3 | 2 |

ตาราง SUBJ

| SUBJ_ID | SUBJ_NAME |
|---|---|
| 1 | Introduction to Computer |
| 2 | E-business |
| 3 | Data Communication |

**2.1 กลุ่มคำสั่งปฏิบัติการปกติกับเซต**

**คำสั่งปฏิบัติการ Union**

แสดงข้อมูลทั้งหมดที่อยู่ในตาราง DEPT1 หรือตาราง DEPT2 ผู้ใช้ต้องพิมพ์คำสั่ง Union ตามหลักไวยกรณ์ คือ

```
result = union dept1 dept2; <Enter>
```

ผลลัพธ์ของคำสั่งเป็นดังนี้

```
MYRDBMS>>result = union deptl dept2;


+--------+----------+
|DEPT_ID |DEPT_NAME |
+--------+----------+
|       1|compsc    |
|       2|math      |
|       3|physics   |
|       4|chem      |
|       5|bio       |
|       6|biochem   |
+--------+----------+
6 row(s) selected
```

ภาพประกอบ ง-7 ตัวอย่างการดำเนินงานคำสั่งปฏิบัติการ Union

**คำสั่งปฏิบัติการ Intersect**

แสดงข้อมูลทั้งหมดที่อยู่ในตาราง DEPT1 และตาราง DEPT2
ผู้ใช้ต้องพิมพ์คำสั่ง Intersect ตามหลักไวยกรณ์ คือ

```
result = intersect dept1 dept2;
```

ผลลัพธ์ของคำสั่งเป็นดังนี้

```
MYRDBMS>>result = intersect deptl dept2;


+--------+----------+
|DEPT_ID |DEPT_NAME |
+--------+----------+
|       1|compsc    |
|       4|chem      |
+--------+----------+
2 row(s) selected
```

ภาพประกอบ ง-8 ตัวอย่างการดำเนินงานคำสั่งปฏิบัติการ Intersect

**คำสั่งปฏิบัติการ Minus**

แสดงข้อมูลทั้งหมดที่อยู่ในตาราง DEPT1 แต่ไม่อยู่ในตาราง DEPT2
ผู้ใช้ต้องพิมพ์คำสั่ง Minus ตามหลักไวยกรณ์ คือ

```
result = minus dept1 dept2; <Enter>
```

ผลลัพธ์ของคำสั่งเป็นดังนี้

```
MYRDBMS>>result = minus deptl dept2;


+--------+----------+
|DEPT_ID |DEPT_NAME |
+--------+----------+
|       2|math      |
|       3|physics   |
+--------+----------+
2 row(s) selected
```

ภาพประกอบ ง-9 ตัวอย่างการดำเนินงานคำสั่งปฏิบัติการ Minus

**คำสั่งปฏิบัติการ Cross**

แสดงข้อมูลที่เป็นการนำข้อมูลที่อยู่ในตาราง DEPT1 มาต่อกับข้อมูลที่อยู่ในตาราง STUDENT

ผู้ใช้ต้องพิมพ์คำสั่ง Cross ตามหลักไวยกรณ์ คือ

```
result = cross dept1 dept2; <Enter>
```

ผลลัพธ์ของคำสั่งเป็นดังนี้

```
MYRDBMS>>result = cross deptl student;

+--------+----------+-------+---------+----------+-----+-----+-------+--------+
|DEPT_ID |DEPT_NAME |STD_ID |STD_NAME |BDATE     |SEX  |HIGH |WEIGHT |DEPT_ID |
+--------+----------+-------+---------+----------+-----+-----+-------+--------+
|       1|compsc    |      1|somsri   | 2/11/2523|F    |  165|     55|       1|
|       1|compsc    |      2|somchai  | 21/2/2522|M    |  180|     70|       1|
|       1|compsc    |      3|sompong  | 15/8/2526|F    |  155|     42|       2|
|       1|compsc    |      4|somwang  |  6/5/2522|M    |  172|     68|       4|
|       2|math      |      1|somsri   | 2/11/2523|F    |  165|     55|       1|
|       2|math      |      2|somchai  | 21/2/2522|M    |  180|     70|       1|
|       2|math      |      3|sompong  | 15/8/2526|F    |  155|     42|       2|
|       2|math      |      4|somwang  |  6/5/2522|M    |  172|     68|       4|
|       3|physics   |      1|somsri   | 2/11/2523|F    |  165|     55|       1|
|       3|physics   |      2|somchai  | 21/2/2522|M    |  180|     70|       1|
|       3|physics   |      3|sompong  | 15/8/2526|F    |  155|     42|       2|
|       3|physics   |      4|somwang  |  6/5/2522|M    |  172|     68|       4|
|       4|chem      |      1|somsri   | 2/11/2523|F    |  165|     55|       1|
|       4|chem      |      2|somchai  | 21/2/2522|M    |  180|     70|       1|
|       4|chem      |      3|sompong  | 15/8/2526|F    |  155|     42|       2|
|       4|chem      |      4|somwang  |  6/5/2522|M    |  172|     68|       4|
+--------+----------+-------+---------+----------+-----+-----+-------+--------+
16 row(s) selected
```

ภาพประกอบ ง-10 ตัวอย่างการดำเนินงานคำสั่งปฏิบัติการ Cross

**2.2 กลุ่มคำสั่งปฏิบัติการพิเศษกับเซต**

**คำสั่งปฏิบัติการ Project**

แสดงข้อมูลรหัส ชื่อ ความสูง และน้ำหนักของนักศึกษา

ผู้ใช้ต้องพิมพ์คำสั่ง Project ตามหลักไวยกรณ์ คือ

```
result = project student [std_id std_name high weight]; <Enter>
```

ผลลัพธ์ของคำสั่งเป็นดังนี้

```
MYRDBMS>>result = project student [std_id std_name high weight];


+-------+---------+-----+-------+
|STD_ID |STD_NAME |HIGH |WEIGHT |
+-------+---------+-----+-------+
|      1|somsri   |  165|     55|
|      2|somchai  |  180|     70|
|      3|sompong  |  155|     42|
|      4|somwang  |  172|     68|
+-------+---------+-----+-------+
4 row(s) selected
```

ภาพประกอบ ง-11 ตัวอย่างการดำเนินงานคำสั่งปฏิบัติการ Project

**คำสั่งปฏิบัติการ Select**

แสดงข้อมูลของนักศึกษาที่มีชื่อว่า “somsri”

ผู้ใช้ต้องพิมพ์คำสั่ง Select ตามหลักไวยกรณ์ คือ

```
result = select student [std_name = “somsri” ];
```

ผลลัพธ์ของคำสั่งเป็นดังนี้

```
MYRDBMS>>result = select student [std_name = "somsri"];


+-------+---------+----------+-----+-----+-------+--------+
|STD_ID |STD_NAME |BDATE     |SEX  |HIGH |WEIGHT |DEPT_ID |
+-------+---------+----------+-----+-----+-------+--------+
|      1|somsri   | 2/11/2523|F    |  165|     55|       1|
+-------+---------+----------+-----+-----+-------+--------+
1 row(s) selected
```

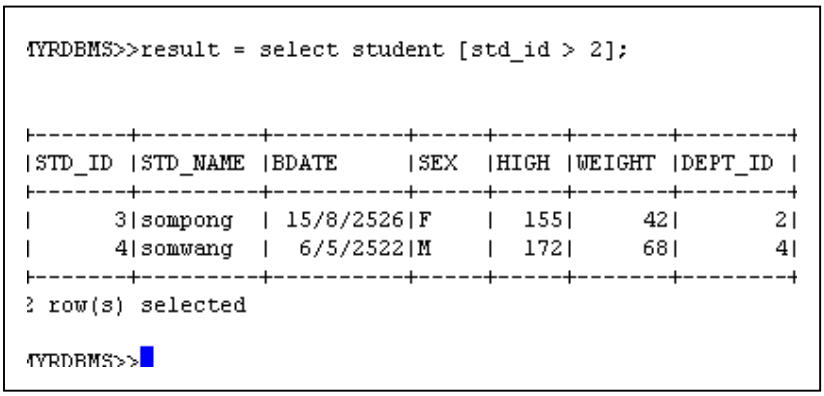
ภาพประกอบ ง-12 ตัวอย่างการดำเนินงานคำสั่งปฏิบัติการ Select เมื่อเปรียบเทียบค่าที่เท่ากัน

แสดงข้อมูลของนักศึกษาที่มีรหัสนักศึกษามากกว่า 2

ผู้ใช้ต้องพิมพ์คำสั่ง Select ตามหลักไวยกรณ์ คือ

```
result = select student [std_id > 2]; <Enter>
```

ผลลัพธ์ของคำสั่งเป็นดังนี้



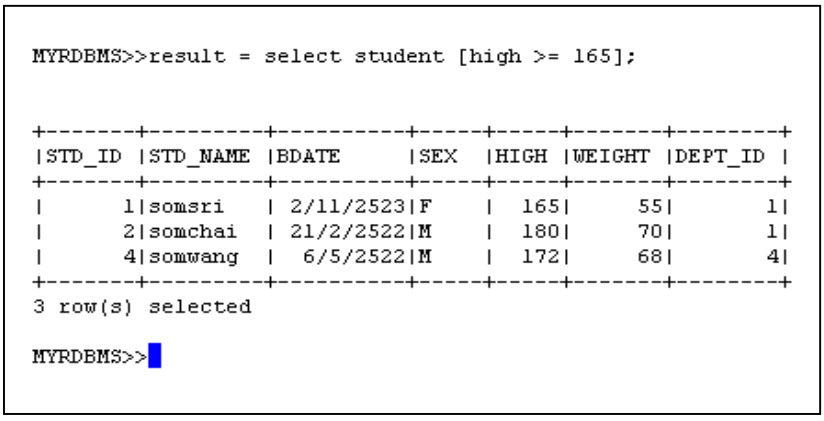
```
MYRDBMS>>result = select student [std_id > 2];

+-------+---------+----------+-----+-----+-------+--------+
|STD_ID |STD_NAME |BDATE     |SEX  |HIGH |WEIGHT |DEPT_ID |
+-------+---------+----------+-----+-----+-------+--------+
|      3|sompong  | 15/8/2526|F    |  155|     42|       2|
|      4|somwang  |  6/5/2522|M    |  172|     68|       4|
+-------+---------+----------+-----+-----+-------+--------+
2 row(s) selected

MYRDBMS>>
```

ภาพประกอบ ง-13 ตัวอย่างการดำเนินงานคำสั่งปฏิบัติการ Select เมื่อเปรียบเทียบค่าที่มากกว่า

แสดงข้อมูลของนักศึกษาที่มีความสูงมากกว่าหรือเท่ากับ 160 เซนติเมตร ผู้ใช้ต้องพิมพ์คำสั่ง Select ตามหลักไวยกรณ์ คือ

```
result = select student [high >= 165]; <Enter>
```

ผลลัพธ์ของคำสั่งเป็นดังนี้



```
MYRDBMS>>result = select student [high >= 165];

+-------+---------+----------+-----+-----+-------+--------+
|STD_ID |STD_NAME |BDATE     |SEX  |HIGH |WEIGHT |DEPT_ID |
+-------+---------+----------+-----+-----+-------+--------+
|      1|somsri   | 2/11/2523|F    |  165|     55|       1|
|      2|somchai  | 21/2/2522|M    |  180|     70|       1|
|      4|somwang  |  6/5/2522|M    |  172|     68|       4|
+-------+---------+----------+-----+-----+-------+--------+
3 row(s) selected

MYRDBMS>>
```

ภาพประกอบ ง-14 ตัวอย่างการดำเนินงานคำสั่งปฏิบัติการ Select เมื่อเปรียบเทียบค่าที่มากกว่าหรือเท่ากับ

แสดงข้อมูลของนักศึกษาที่มีน้ำหนักน้อยกว่า 50 กิโลกรัม ผู้ใช้ต้องพิมพ์คำสั่ง Select ตามหลักไวยกรณ์ คือ

```
result = select student [weight < 55 ]; <Enter>
```

ผลลัพธ์ของคำสั่งเป็นดังนี้

```
MYRDBMS>>result = select student [weight < 55 ];


+-------+---------+----------+-----+-----+-------+--------+
|STD_ID |STD_NAME |BDATE     |SEX  |HIGH |WEIGHT |DEPT_ID |
+-------+---------+----------+-----+-----+-------+--------+
|      3|sompong  | 15/8/2526|F    |  155|     42|       2|
+-------+---------+----------+-----+-----+-------+--------+
1 row(s) selected

MYRDBMS>>
```

ภาพประกอบ ง-15 ตัวอย่างการดำเนินงานคำสั่งปฏิบัติการ Select เมื่อเปรียบเทียบค่าที่น้อยกว่า

แสดงข้อมูลของนักศึกษาที่มีความสูงน้อยกว่าหรือเท่ากับ 180 เซนติเมตร ผู้ใช้ต้องพิมพ์คำสั่ง Select ตามหลักไวยกรณ์ คือ

```
result = select student [high <= 180]; <Enter>
```

ผลลัพธ์ของคำสั่งเป็นดังนี้

```
MYRDBMS>>result = select student [high <= 180];


+-------+---------+----------+-----+-----+-------+--------+
|STD_ID |STD_NAME |BDATE     |SEX  |HIGH |WEIGHT |DEPT_ID |
+-------+---------+----------+-----+-----+-------+--------+
|      1|somsri   | 2/11/2523|F    |  165|     55|       1|
|      2|somchai  | 21/2/2522|M    |  180|     70|       1|
|      3|sompong  | 15/8/2526|F    |  155|     42|       2|
|      4|somwang  |  6/5/2522|M    |  172|     68|       4|
+-------+---------+----------+-----+-----+-------+--------+
4 row(s) selected

MYRDBMS>>
```

ภาพประกอบ ง-16 ตัวอย่างการดำเนินงานคำสั่งปฏิบัติการ Select เมื่อเปรียบเทียบค่าที่น้อยกว่าหรือเท่ากับ

แสดงข้อมูลของนักศึกษาที่มีเป็นเพศหญิง
ผู้ใช้ต้องพิมพ์คำสั่ง Select ตามหลักไวยกรณ์ คือ

```
result = select student [sex != 'M']; <Enter>
```

ผลลัพธ์ของคำสั่งเป็นดังนี้

```
MYRDBMS>>result = select student [sex != 'M' ];


+-------+---------+----------+-----+-----+-------+--------+
|STD_ID |STD_NAME |BDATE     |SEX  |HIGH |WEIGHT |DEPT_ID |
+-------+---------+----------+-----+-----+-------+--------+
|      1|somsri   | 2/11/2523|F    |  165|     55|       1|
|      3|sompong  | 15/8/2526|F    |  155|     42|       2|
+-------+---------+----------+-----+-----+-------+--------+
2 row(s) selected

MYRDBMS>>
```

ภาพประกอบ ง-17 ตัวอย่างการดำเนินงานคำสั่งปฏิบัติการ Select เมื่อเปรียบเทียบค่าที่ไม่เท่ากับด้วยเครื่องหมาย “!=”

หรือผู้ใช้พิมพ์คำสั่ง Select ตามหลักไวยกรณ์ คือ

```
result = select student [sex <> 'M']; <Enter>
```

ผลลัพธ์ของคำสั่งเป็นดังนี้

```
MYRDBMS>>result = select student [sex <> 'M'];


+-------+---------+----------+-----+-----+-------+--------+
|STD_ID |STD_NAME |BDATE     |SEX  |HIGH |WEIGHT |DEPT_ID |
+-------+---------+----------+-----+-----+-------+--------+
|      1|somsri   | 2/11/2523|F    |  165|     55|       1|
|      3|sompong  | 15/8/2526|F    |  155|     42|       2|
+-------+---------+----------+-----+-----+-------+--------+
2 row(s) selected

MYRDBMS>>
```

ภาพประกอบ ง-18 ตัวอย่างการดำเนินงานคำสั่งปฏิบัติการ Select เมื่อเปรียบเทียบค่าที่ไม่เท่ากับด้วยเครื่องหมาย “<>”

**คำสั่งปฏิบัติการ Join**

แสดงข้อมูลของนักศึกษาและภาควิชาที่นักศึกษาสังกัดอยู่

ผู้ใช้ต้องพิมพ์คำสั่ง Join ตามหลักไวยกรณ์ คือ

```
result = join dept1 student [dept_id = dept_id ]; <Enter>
```

ผลลัพธ์ของคำสั่งเป็นดังนี้

```
MYRDBMS>>result = join deptl student [dept_id = dept_id];


+--------+----------+-------+---------+----------+-----+-----+-------+--------+
|DEPT_ID |DEPT_NAME |STD_ID |STD_NAME |BDATE     |SEX  |HIGH |WEIGHT |DEPT_ID |
+--------+----------+-------+---------+----------+-----+-----+-------+--------+
|       1|compsc    |      1|somsri   | 2/11/2523|F    |  165|     55|       1|
|       1|compsc    |      2|somchai  | 21/2/2522|M    |  180|     70|       1|
|       2|math      |      3|sompong  | 15/8/2526|F    |  155|     42|       2|
|       4|chem      |      4|somwang  |  6/5/2522|M    |  172|     68|       4|
+--------+----------+-------+---------+----------+-----+-----+-------+--------+
4 row(s) selected
```

ภาพประกอบ ง-19 ตัวอย่างการดำเนินงานคำสั่งปฏิบัติการ Join

**คำสั่งปฏิบัติการ Divide**

แสดงข้อมูลรหัสนักศึกษาที่ลงทะเบียนเรียนวิชา “Introduction to Computer” และวิชา “E-business”

ผู้ใช้ต้องพิมพ์คำสั่ง Divide ตามหลักไวยกรณ์ คือ

```
result = divide std_subj subj_id; <Enter>
```

ผลลัพธ์ของคำสั่งเป็นดังนี้

```
MYRDBMS>>result = divide std_subj subj_id;
+-----------+
|STD_ID     |
+-----------|
|3          |
+-----------+
l row(s) selected
```

ภาพประกอบ ง-20 ตัวอย่างการดำเนินงานคำสั่งปฏิบัติการ Divide

**คำสั่งปฏิบัติการ Unique**

แสดงข้อมูลที่อยู่ในตาราง DEPT2 โดยถ้ามีแถวข้อมูลแนวนอนที่ซ้ำกันให้เลือกเพียงหนึ่งแถวข้อมูลแนวนอน

ผู้ใช้ต้องพิมพ์คำสั่ง Unique ตามหลักไวยกรณ์ คือ

```
result = unique dept2; <Enter>
```

ผลลัพธ์ของคำสั่งเป็นดังนี้

```
MYRDBMS>>result = unique dept2;


+---------+-----------+
|DEPT_ID  |DEPT_NAME  |
+---------+-----------+
|        1|compsc     |
|        5|bio        |
|        6|biochem    |
|        4|chem       |
+---------+-----------+
4 row(s) selected
```

ภาพประกอบ ง-21 ตัวอย่างการดำเนินงานคำสั่งปฏิบัติการ Unique

**คำสั่งปฏิบัติการ Asce**

แสดงข้อมูลของนักศึกษา โดยเรียงน้ำหนักของนักศึกษาจากค่าน้อยไปหาค่ามาก

ผู้ใช้ต้องพิมพ์คำสั่ง Dsce ตามหลักไวยกรณ์ คือ

```
result = asce student [weight]; <Enter>
```

ผลลัพธ์ของคำสั่งเป็นดังนี้

```
MYRDBMS>>result = asce student[weight];


+-------+---------+----------+-----+-----+-------+---------+
|STD_ID |STD_NAME |BDATE     |SEX  |HIGH |WEIGHT |DEPT_ID  |
+-------+---------+----------+-----+-----+-------+---------+
|      3|sompong  | 15/8/2526|F    |  155|     42|        2|
|      1|somsri   | 2/11/2523|F    |  165|     55|        1|
|      4|somwang  |  6/5/2522|M    |  172|     68|        4|
|      2|somchai  | 21/2/2522|M    |  180|     70|        1|
+-------+---------+----------+-----+-----+-------+---------+
4 row(s) selected
```

ภาพประกอบ ง-22 ตัวอย่างการดำเนินงานคำสั่งปฏิบัติการ Asce

**คำสั่งปฏิบัติการ Dsce**

แสดงข้อมูลของนักศึกษา โดยเรียงส่วนสูงของนักศึกษาจากค่ามากไปหาค่าน้อย
ผู้ใช้ต้องพิมพ์คำสั่ง Dsce ตามหลักไวยกรณ์ คือ

```
result = dsce student [high]; <Enter>
```

ผลลัพธ์ของคำสั่งเป็นดังนี้

```
MYRDBMS>>result = dsce student[high];

+-------+---------+----------+-----+-----+-------+--------+
|STD_ID |STD_NAME |BDATE     |SEX  |HIGH |WEIGHT |DEPT_ID |
+-------+---------+----------+-----+-----+-------+--------+
|      2|somchai  | 21/2/2522|M    |  180|     70|       1|
|      4|somwang  |  6/5/2522|M    |  172|     68|       4|
|      1|somsri   | 2/11/2523|F    |  165|     55|       1|
|      3|sompong  | 15/8/2526|F    |  155|     42|       2|
+-------+---------+----------+-----+-----+-------+--------+
4 row(s) selected
```

ภาพประกอบ ง-23 ตัวอย่างการดำเนินงานคำสั่งปฏิบัติการ Dsce

## 2.3 กลุ่มคำสั่งปฏิบัติการแบบฟังก์ชัน

**คำสั่งปฏิบัติการ Count**

แสดงจำนวนแถวข้อมูลแนวนอนที่อยู่ในตาราง STUDENT
ผู้ใช้ต้องพิมพ์คำสั่ง Count ตามหลักไวยกรณ์ คือ

```
result = count student ; <Enter>
```

ผลลัพธ์ของคำสั่งเป็นดังนี้

```
MYRDBMS>>result = count student;

+--------------+
|COUNT(STUDENT)|
+--------------+
|4             |
+--------------+
```

ภาพประกอบ ง-24 ตัวอย่างการดำเนินงานคำสั่งปฏิบัติการ Count

**คำสั่งปฏิบัติการ Max**

แสดงส่วนสูงของนักศึกษาที่มีค่ามากที่สุด

ผู้ใช้ต้องพิมพ์คำสั่ง Max ตามหลักไวยกรณ์ คือ

```
result = max student[high] ; <Enter>
```

ผลลัพธ์ของคำสั่งเป็นดังนี้

```
MYRDBMS>>result = max student[high];

+----------+
|MAX(HIGH) |
+----------+
|180       |
+----------+
```

ภาพประกอบ ง-25 ตัวอย่างการดำเนินงานคำสั่งปฏิบัติการ Max

**คำสั่งปฏิบัติการ Min**

แสดงน้ำหนักของนักศึกษาที่มีค่าน้อยที่สุด

ผู้ใช้ต้องพิมพ์คำสั่ง Min ตามหลักไวยกรณ์ คือ

```
result = min student[weight] ; <Enter>
```

ผลลัพธ์ของคำสั่งเป็นดังนี้

```
MYRDBMS>>result = min student[weight];

+-----------+
|MIN(WEIGHT)|
+-----------+
|42         |
+-----------+
```

ภาพประกอบ ง-26 ตัวอย่างการดำเนินงานคำสั่งปฏิบัติการ Min

**คำสั่งปฏิบัติการ Sum**

แสดงผลรวมของส่วนสูงของนักศึกษา

ผู้ใช้ต้องพิมพ์คำสั่ง sum ตามหลักไวยกรณ์ คือ

```
result = sum student[high] ; <Enter>
```

ผลลัพธ์ของคำสั่งเป็นดังนี้

```
MYRDBMS>>result = sum student[high];

+----------+
|SUM(HIGH) |
+----------+
|672       |
+----------+
```

ภาพประกอบ ง-27 ตัวอย่างการดำเนินงานคำสั่งปฏิบัติการ Sum

**คำสั่งปฏิบัติการ Avg**

แสดงค่าเฉลี่ยของน้ำหนักของนักศึกษา

ผู้ใช้ต้องพิมพ์คำสั่ง Avg ตามหลักไวยกรณ์ คือ

```
result = avg student[high] ; <Enter>
```

ผลลัพธ์ของคำสั่งเป็นดังนี้

```
MYRDBMS>>result = avg student[weight];

+-----------+
|AVG(WEIGHT)|
+-----------+
|58.75      |
+-----------+
```

ภาพประกอบ ง-28 ตัวอย่างการดำเนินงานคำสั่งปฏิบัติการ Avg

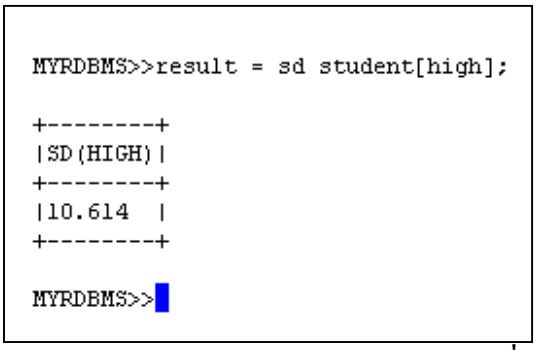
**คำสั่งปฏิบัติการ Sd**

แสดงค่าเฉลี่ยของน้ำหนักของนักศึกษา

ผู้ใช้ต้องพิมพ์คำสั่ง Sd ตามหลักไวยกรณ์ คือ

```
result = sd student[high] ; <Enter>
```

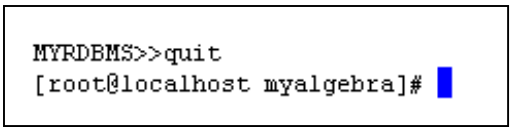
ผลลัพธ์ของคำสั่งเป็นดังนี้




ภาพประกอบ ง-29 ตัวอย่างการดำเนินงานคำสั่งปฏิบัติการ Sd

### 3. การออกจากระบบ

เมื่อต้องการออกจากระบบหรือหยุดการทำงานของโปรแกรม จะต้องใช้คำสั่ง “quit” เพื่อให้ระบบทำการปิดแฟ้มข้อมูลที่เปิดใช้งานในระบบด้วย มิฉะนั้นแฟ้มข้อมูลจะไม่สามารถใช้งานได้อีกในการทำงานระบบครั้งต่อไป




ภาพประกอบ ง-30 การออกจากระบบ