

**Anugrah Memorial College, Katari hill road, Gaya**  
**Prepared by: Sanjiw Kumar**

**FOXPRO**

**DBMS** stand for Data Base Management System. In this system you can create files, reports, labels etc. The database is the collection of records related to person. The DBMS packages are Dbase III plus, Dbase IV, FoxBASE, FoxPro etc.

**FoxPro** is very powerful Database Management System. Its integrate development Environment allows programmer's language. It is both an interpreter and true compiler. As interpreted language, it translates each command into computer instructions, as it is executes the translation time, most possibly during loops.

A compiler improves performance by translating command to machine instructions just once. This compiled programs run faster than interpreted codes.

The main screen combines a menu bar across the top line with a command prompt environment based on the command window. FoxPro opens the command window at the beginning of each session. Commands entered into it execute immediately.

FoxPro is a database management system. It means that using this software we can handle the data in different way. The activity include in DBMS system are

- 1] Create a database or format structure to store data.
- 2] Add records in database.
- 3] Modify records, Edit records.
- 4] Search particular records.
- 5] Arrange records in particular order (ascending or descending order).
- 6] Delete records
- 7] Delete Database or modify structure.
- 8] Prepare Reports and Labels.

The three important terms are

- 1] **Field** :- It is the unit or information related to person. Or it is a heading of column in database.
- 2] **Record** :- It is the collection of fields or a complete row of table is called record.
- 3] **Database** :- The collection of records is called file.

**1] Creating File in FoxPro**

Create: To create file in foxpro use command CREATE command window with file name.

Syntax:- *\_Create <filename>*

e.g.

### Create data

A structure dialog box appear on screen

Here enter field names, there datatype (datatype means type of data you allowed for that field for e.g. number or character or date) enter width of field (space allowed) and if necessary enter decimal number.

After this select OK button or press Ctrl+W to save and exit form structure box, a message display to "Input record now (y/n)" press y to add records in append window.

Add some records in append window and to close and save data in file press Ctrl +W again.

### Data type available in FoxPro

Data type	Default width	Maximum Width
Character	10	254
Numeric	10	20
Float	10	20
Date	08	08
Logical	01	01
Memo	10 in dbf file	unlimited bytes
General	10 in dbf file	unlimited bytes

In FoxPro there are different data types are available. These are

- 1] **Character** :- Allowed alphabets (A to Z, a to z) number (0 to 9) and special character and space.
- 2] **Numeric** :- Allowed digits (0 to 9) and decimal point also contain + or - sign.
- 3] **Float** :- Same as Numeric.
- 4] **Date**:- the date data type use to define or enter date value.

5] **Memo** :- The memo data type is same as character data type having unlimited width.

6] **General** :- it is use for OLE object.

### Rules for naming field

- ✓ 1] Field name must start with character or underscore (\_) character.
- ✓ 2] Field name can be up to 10 char.
- ✓ 3] Space is not allowed in field name.
- ✓ 4] Duplicate field name is not allowed.

### 2] Use Command:

This command is use to open an existing file and also to close current file.

**Syntax** :- Use [File name]

e.g. to open data file

Use Data

### 3] Append

This command is use to add new records

Syntax

*Append [blank]*

*[from <filename>]*

*[field <field list>]*

*[for <expL>]*

e.g.

1] to add records in append window

use data

Append

2] To add blank record without opening append window

append blank

3] To add new records in a current file from file name "student"

append from student

4] To add only those records having city "shegaon" of "student" file  
append from student for city = "shegaon"

4] **Browse:** This command is most popular or useful in FoxPro which let you to add new records, delete records or modify records in browse window.

**Syntax:-**

Browse [Scope]

[Fields <fieldlist>]

[For <expL>]

[While<expL>]

[noappend] [nomodify]

[nodelete] [noedit]

e.g

1] To open browse window

Browse

2] To browse only fields Title author and cost only

browse fields Title,Author,Cost

3] To browse only records having cost greater than 300

browse all for cost >300

4] To browse records but not able to add, modify or delete records in browse window

browse all noappend nodelete nomodify

5] **List :-** this command use to list records on screen, file or to send out put to printer

**Syntax:-** List [off] [scope]

[field <field list>]

[for<expL>]

[While<expL>]

e.g.

1] To list all records

list

☞ 2] To list records without record number  
List off

☞ 3] To list record only 1 to 3  
go top

list next 3

☞ 4] To list only fields title and cost  
List fields title,cost

☞ 5] To list only record of author E. Balguruswami  
List all for author="E.balguruswami"

6] To list only title and author of title FoxPro  
List fields title, author for title = "FoxPro"

5] **Display** :- this command use to Display records on screen, file or to send out put to printer

### Syntax:-

Display [off] [scope]

[field <field list>]

[for<expL>]

[While<expL>]

e.g.

Display: to display only current record

☞ To Display all records  
Display all

☞ To Display records without record number  
Display off

☞ To Display record only 1 to 3  
go top

Display next 3

☞ To Display only fields title and cost  
Display fields title,cost

☞ To Display only record of author E. Balguruswami  
Display all for author="E.balguruswami"

☞ To Display only title and author of title FoxPro  
Display fields title, author for title = "FoxPro"

**6] Delete :** - This command is use to mark records for delete.

**Syntax:-** Delete [scope]

[field <field list>]

[for<expL>]

[While<expL>]

e.g.

☞ To Delete all records  
Delete all

☞ To Delete record only 1 to 3  
go top  
Delete next 3

☞ To Delete only record of author E. Balguruswami  
Delete all for author="E.balguruswami"

**7] Recall:-** This command is use to recall records marked for delete.

**Syntax :-** Recall [scope]

[field <field list>]

[for<expL>]

[While<expL>]

e.g.

☞ To Recall all records  
Use stud  
Delete for city="Shegaon"  
Recall all

☞ To Recall record only 1 to 3  
Use stud  
Delete all  
go top  
Recall next 3

☞ To Recall only record of author E. Balguruswami  
Recall all for author = "E.balguruswami"

8] **Pack:-** This command is use to delete records permanently marked for delete.

**Syntax:-** Pack

e.g. delete all record.

use stud & pack

9] **Zap:-** This command is use to delete all record permanently of currently open .dbf file.

e.g

To delete all record of data file

use data

zap

10] **Go to:-** This command lets you to move record pointer in a dbf file.

**Syntax:-**

e.g

use stud

goto 3

11] **Locate** :- This command is use to search the record sequentially that matches given expression.

Locate sequentially search the current table for the first record that matches a logical expression. The table doesn't have to be indexed.

If locate find the record it display it's Record no else place the record pointer to EOF.


**Syntax:-**

Locate [scope]


For <expL>

[While<expL>]

e.g

 To locate records having city shegaon  
use stud

Locate all for city = "shegaon"

 To locate all record between 3000 to 6000  
Locate all for salary > = 3000 .and. salary<=6000

12] **Continue** :- Continue command is used after LOCATE succeeds in finding a record, to continue the LOCATE operation. CONTINUE moves the record pointer to the next record for which the logical expression specified in the previous LOCATE evaluates to true <.T.>

13] **Sort** :- Sort command sort the current table and outputs the sorted data to a new table.

**Syntax:-**

SORT TO <file> ON <field1> [/A | /D] ,

<field2> [/A | /D]

[ascending] [descending]

[scope]

[for <expL>]

[while<expL>]



<file> when a table is sorted a new table is created

<on> <field1> you must include a name of a field

[/A | /D] for each field you include in the sort you can specify an ascending or descending sort order.


ASCENDING | DESCENDING

You can specify a sort order for all sort fields not followed by /A or /D


<scope>

The scope clauses are: All, NEXT <expN>, REOCD <expN> and REST


e.g.

 To sort the current table on field name and store it to temp table  
Use stud

Sort to temp on name /a

 To sort the current table on field name and store it to temp table but in descending order  
Use stud

Sort to temp on name desc

 To sort current table on Title and include only selected fields to it  
Use lib

Sort to abc on title fields title, author, cost

**14] Index:-** This command is used to arrange the records in ascending or descending order. There are two types of indexing compound indexing (.cdx) containing multiple entries called tag and simple index (idx) containing single index entry.

### Syntax

Index on <field exp>

[to <idx file>]/[tag <tag name>]

[for <expL>]

[compact] [ascending | descending]

<field exp> : include fields on which you want to index file (not allowed different data type fields if u want to use then first convert its data type using function.)

e.g

use lib

index on name,dtoc(dofp) tag a additive

for <expL> Only records that satisfy the filter expression <expL> are available for display and access.

Compact create a compact .idx file.

ASCENDING | DESCENDING .cdx files can be built in ascending or descending order. By default tag are created in ascending order.

e.g

to index file on name in ascending order  
use abc

index on name tag a

15] **FIND:-** (index query for a single record)

Search an index table. Find moves the record pointer to the first record in the table whose index key matches the character expressing <expC>.

If the matching record is found RECNO() return it's record number, FOUND() return .T. and EOF() return .F.

If SET NEAR is ON and find is unsuccessful, the record pointer is positioned immediately after the closest matching record.

e.g

use lib

index on Title tag a additive

find "FoxPro for Windows"

16] **SEEK:-** SEEK command is similar to FIND search record in index table. In this expression is allowed.

e.g.

use lib

index on cost tag a

c=234

seek c

## 17] SET COMMANDS

a) SET STATUS ON | OFF :  
Display or remove Status bar.

b) SET CLOCK ON | OFF :

SET CLOCK TO [ <ROW> ,<COLUMN>]

Determine whether or not FoxPro displays the system date and specify its location. On the screen. (total number on screen is 24 and column is 80 )

e.g.

set clock on

set clock to 12,30

c) SET CENTURY ON | OFF :

Determine whether or not FoxPro displays the century portion of date expression in four digits or two digits.

d) SET BELL ON | OFF

Turn the computer bell on or off and set the bell attributes.

syntax

SET BELL ON | OFF

SET BELL [<frequency>,<duration>]

Frequency is between 19 to 10000 hertz

And duration is between 1 to 19

e.g.

SET BELL ON

SET BELL TO 12000 , 12

e) SET DELETED ON | OFF

Specify whether or not FoxPro processes records marked for deletion Or available for use in other commands

f) SET TALK ON | OFF

Determine whether or not FoxPro displays command results

g) SET ESCAPE ON | OFF

Determine whether or not pressing the Escape key interrupts program and command execution.

h) SET HEADING ON | OFF

Set Heading specifies whether a field name is displayed as a column heading output of average, calculate, display, list and sum.

i) SET NEAR ON | OFF

Determine whether the record pointer is positioned after FIND or SEEK unsuccessfully search for a record.

j) SET SAFETY ON | OFF

Determines whether or not FoxPro displays a warning before overwriting an existing file.

k) SET EXACT ON | OFF

Specifies rules used when comparing strings of different lengths.

l) SET DATE [TO] AMERICAN | ANSI |  
BRITISH | FRENCH | GERMAN

| ITALIAN | JAPAN | USA | MDY

| DMY | YMD

Use this command to set the format used to display date expressions. Here are the settings and the resulting date expressions.

Resulting date format

Setting	Format
American	mm/dd/yy
Ansi	yy.mm.dd
Italian	dd-mm-yy
German	dd.mm.yy
British / French	dd/mm/yy
Japan	yy/mm/dd
Usa	mm-dd-yy
Mdy	mm-dd-yy
Dmy	dd-mm-yy
Ymd	yy/mm/dd

#### SET DECIMAL TO <expN>

Specify the decimal places display in numeric used to display the result of division, multiplication and trigonometric and financial function.

e.g.

set decimal to 2

#### m) SET CARRY ON/OFF

Determine whether or not foxpro carry forward the entire record from current record created by APPEND command.

#### n) SET CARRY TO <field>

Specifies the fields from which you want to carry forward the data to the new record.

e.g.

use stud

set carry to city,course

append

#### o) SET CONFIRM ON/OFF

Specifies whether or not enter or tab key pressed to exit an input field and move to the next field.

By default confirm is *OFF*

p) SET DEFAULT TO <path>

Specifies the default drive & directory for working .

e.g.

*set default to a:\main*

q) SET FILTER TO

Specifies a condition for the current database file. This display only those records which satisfied the given condition.

e.g.

*use stud*

*set filter to course="ADCSSAA"*

*browse*

*tip:- to remove the filter give command SET FILTER TO ALL*

r) SET INDEX TO

Open one or more index files for use with current database file. You can open both .IDX or .CDX files with this command.

e.g.

*use stud*

*index on name to name*

*index on rollno to roll*

*set index to name*

s) SET ORDER TO

Select a specified index file or tag as the controlling index file/tag for the current or specified file.

e.g.

*use stud*

*index on name tag name*

*index on rollno tag roll*

*set order to name*

t) SET MESSAGE TO <expN>

Specifies the message to row location created by user.

u) SET RELATION TO

Create the relation with more than one file using this command.

e.g.

suppose the two files are created with name & fields as

emp_c	emp_v
code	code
name	post
dob	salary
doj	city

*tip:- you must remember that you must take one field command in both file. Here is **code**.*

Now set the relation as:-

*select 1*

*use emp\_c*

*index on code to code*

*select 2*

*use emp\_v*

*index on code to code1*

*select 1*

*set relation to code into emp\_v*

*browse fields code,name,emp\_v.post,emp\_v.salary*

v) **SET UNIQUE ON/OFF**

Specifies whether or not the duplicate records are displayed or not in *index file*.

w) **SET PROCEDURE TO**

Opens the procedure file. Only one procedure file is opened at a time.

## DATE FUNCTIONS

1) **DATE()**:- Returns the current system date, which is controlled by the operating system.

**Syntax & e.g. :-**

?DATE( )

Returns date:-

02/14/01

*tip:- if century is on then year displayed in yyyy format.*

2) **DAY()** :- Returns the numeric day of the month for a given date expression.

DAY( ) returns a number from 1 through 31.

**Syntax :-**

DAY(<expD>)

e.g. ?DAY({01/11/99})

Returns value :- 10

3) **MONTH()**:- Return the numeric month of given date expression.

**Syntax:-**

Month(<expD>)

e.g.

?MONTH({10/09/1999})

Return value :- 10

4) **CMONTH()**:- Return the character month of given date expression.

**Syntax:-**

CMONTH(<expD>)

e.g. ?CMONTH({09/03/79})

Return value:- September

*Tip :- you can also write the character text to this date as*

?My birth day will come in the month of ,cmonth({09/03/1979})

*Return :- My birth day will come in the month of September*

5) **DOW()**:- Returns the numeric day of week from given date expression.

The value returned by DOW( ) ranges from 1 (Sunday) through 7 (Saturday).

**Syntax:-**

DOW(<expD>)

e.g. ?DOW({date()})



return value :- 1.  
e.g. ?Dow({02/14/2001})  
Return value :- 4

**6) CDOW():-** Return the character day of week from given date expression.

**Syntax:-**

*CDOW(<expD>)*

e.g. ?CDOW({02/14/2001})

Return :- Wednesday

**7) YEAR():-** Return the year from given date expression. SET CENTURY ON/OFF command will not affect this function it always displayed in YYYY format.

**Syntax:-**

*YEAR(<expD>)*

e.g. ?YEAR({01/01/99})

Return:- 1999

**8) CTOD():-** Return the a character expression to a date expression.

**Syntax:-**

*CTOD(<expC>)*

e.g.. ?CTOD("10/01/1999")

Returns:- 10/01/1999

*Tip:- CTOD( ), the character to date function, returns a date type value from a character expression.*

**9) DTOC():-** Convert the date expression into character expression.

**Syntax:-** *DTOC(<expD>)*

e.g. ?DTOC({10/01/1999})

Return :- 10/01/1999

**10) DTOS():-** Returns a character-string date in the format YYYYMMDD from a specified date expression.

**Syntax:-** *DTOS(<expD>)*

e.g. ?DTOS({10/09/1999})

Returns:- 19991009

**11) DMY():-** Returns a specified date expression in Day Month Year format.

**Syntax:-** *DMY(<expD>)*

e.g. ?DMY({02/10/2001})

Returns:- 10 February 01

*Tip:- if SET CENTURY OFF then year displayed in yy format otherwise in YYYY format.*

**12) MDY():-** Returns the specified date expression in Month Day Year format.

**Syntax :-** *MDY(<expD>)*

e.g. ?MDY({02/14/2001})

Returns:- February 14, 2001

## NUMERIC FUNCTIONS

1) **MOD()**:- Returns the remainder obtained by dividing a numeric expression by another numeric expression.

**Syntax** :-  $MOD(<expN1>, <expN2>)$

e.g. ?MOD(10,3)

Return :- 1

*Tip:- MOD is identical to %*

2) **ABS()**:- Returns the absolute value of the specified numeric expression.

**Syntax** :-  $ABS(<expN>)$

e.g. ?ABS(-129.90)

Returns: 129.90

3) **ACOS()**:- Returns the arc cosine of a specified numeric expression in radian.

**Syntax** :-  $ACOS(<expN>)$

e.g. ?ACOS(0.10)

Returns:- 1.47

*Tip:- The value of <expN> can range from +1 through -1. The value returned by ACOS( ) ranges from 0 through pi ( 3.141592).*

4) **ASIN()**:- This trigonometric function returns in radians the arc sine of the numeric expression <expN>.

**Syntax** :-  $ASIN(<expN>)$

e.g. ?ASIN(-1)

Returns:- -1.57

*Tip:- The value of <expN> can range from +1 through -1, and the value returned by ASIN( ) can range from -pi/2 through +pi/2 ( -1.57079 to 1.57079).*

5) **ATAN()**:- This trigonometric function returns in radians the arc tangent of a numeric expression.

**Syntax** :-  $ATAN(<expN>)$

e.g. ?ATAN(1.3)

Returns:- 0.92

*Tip:- the value returned by ATAN( ) can range from -pi/2 through +pi/2 ( -1.57079 to 1.57079). The numeric value returned by ATAN( ) is in radians.*

6) **COS()**:- Returns the cosine of a numeric expression.

**Syntax**:-  $COS(<expN>)$

e.g. ?cos(45)

Returns 0.53

*Tip:- the value returned by COS( ) ranges between -1 and 1. <expN> is given in radians.*

7) **DTOR()**:- Converts the value of a numeric expression given in degrees to an equivalent value in radians.

**Syntax** :-  $DTOR(<expN>)$

e.g. ?DTOR(180)

Returns:- 3.14

8) **EXP()** :- Returns the value of  $e^x$  where x is a specified numeric expression. The value of e, the base of natural logarithms, is approximately 2.71828.

**Syntax:-** EXP(<expN>)

e.g. ?EXP(2)

Returns:- 7.39

9) **FLOOR()**:- Returns the nearest integer that is less than or equal to the specified numeric expression.

**Syntax:-** FLOOR(<expN>)

e.g:- ?FLOOR(19,90)

Returns:- 19

e.g. ?FLOOR(-19.99)

Return:- -20

*TIPS:- FLOOR( ) rounds a positive number with a fractional portion down to the integer portion of the number, and rounds a negative number with a fractional portion down to the next lowest integer.*

10) **INT()**:- Evaluates a numeric expression and returns the integer portion of the expression.

**Syntax :-** INT(<expN>)

e.g. :- ?INT(20.99)

Returns:- 19

11) **MAX()**:- Returns the expression with the highest ASCII or numeric value or the latest date from a list of character, numeric or date expressions.

**Syntax :-** MAX(<expr1>, <expr2>[, <expr3> ... ])

e.g. ?max(12,22,10)

Returns:- 22

12) **MIN()**:- Returns the expression with the lowest ASCII or numeric value or the earliest date in a list of character, numeric or date expressions.

**Syntax :-** MIN(<expr1>, <expr2>[, <expr3> ...])

e.g. ?min(10,9,8)

Returns:- 8

13) **ROUND()**:- Return a numeric expression rounded to a specified number of decimal places.

**Syntax:-** ROUND(<expN1>, <expN2>)

e.g. ?ROUND(19.9956,2)

Return:- 20.00

*Tip :- if the number is greater than 5, then 1 is added to the previous number.*

14) **RTOD()**:- Converts a numeric expression that represents radians to an equivalent

value in degrees.

**Syntax :-** *RTOD(<expN>)*

e.g. ?RTOD(3.1416)

Returns:- 180.00

15) **SQRT()**:- Returns the square root of the specified numeric expression.

**Syntax :-** *SQRT(<expN>)*

e.g. ?SQRT(121)

Return :- 11

16) **VAL()**:- Returns a numeric expression from a specified character expression composed of numbers.

**Syntax:-** *VAL(<expC>)*

e.g. a='12'

?VAL(a)

Return:- 12.00

## CHARACTER FUNCTIONS

1) **ALLTRIM()**:- Returns the specified character expression with leading and trailing blanks removed.

**Syntax:-** *ALLTRIM(<expC>)*

e.g. ?""HELLO FRIEND",ALLTRIM(" THIS IS TEST ", " example")

Return:- HELLO FRIEND THIS IS TEST example.

*Tip:- ALLTRIM( ) removes leading and trailing blanks from the specified character expression and returns the trimmed expression as a character string.*

2) **ASC()**:- Returns the ASCII code for the leftmost character in a character expression.

**Syntax :-** *ASC(<expC>)*

e.g. ?ASC('a')

Returns:- 97

*Tip:-Every character has a unique ASCII value in the range from 0 to 255.*

3) **AT()**:- Returns the beginning numeric position of the first occurrence of a character expression or field within another character expression or field, counting from the leftmost character.

**Syntax:-** *AT(<expC1>, <expC2>[, <expN>])*

Here ,<expC1> - is the character whose position you want to see.

<expC2>- is the text in which you want to search.

<expN>- is the number of occurrence.

e.g.- ?at('A','DATA POINT',2)

Return:- 4

4) **CHR()**:- Returns the character associated with the specified numeric ASCII code.

**Syntax :-** *CHR(<expN>)*

e.g. ?CHR(97)

Return:- a

5) **ISALPHA()**:- Returns true (.T.) if the leftmost character in the specified character expression is an alphabetic character; otherwise, false (.F.) is returned.

**Syntax :-** *ISALPHA(<expC>)*

e.g. ?ISALPHA("12data point")

Returns:- .F.

7) **ISDIGIT()**:- Returns true (.T.) if the leftmost character of the specified character expression is a digit (09); otherwise, false (.F.) is returned.

**Syntax:-** *ISDIGIT(<expC>)*

e.g. ?ISDIGIT("12 data ")

Return:- .T.

8) **ISLOWER()**:- Returns true (.T.) if the leftmost character in the specified character expression is a lower-case alphabetic character.

**Syntax :-** *ISLOWER(<expC>)*

e.g. ?ISLOWER("DATA POIN")

Return:- .F.

9) **ISUPPER()**:- Returns true (.T.) if the first character in a character expression is an upper-case alphabetic character.

**Syntax:-** *ISUPPER(<expC>)*

e.g. ?ISUPPER("DATA POINT")

Return:- .T.

10) **LEFT()**:- Returns a specified number of characters from a character expression, starting with the leftmost character.

**Syntax:-** *LEFT(<expC>, <expN>)*

e.g. ?LEFT(" data point",3)

Return:- dat

11) **LEN()**:- Returns the number of characters in a character expression.

**Syntax :-** *LEN(<expC>)*

e.g. ?LEN("data point")

Return:- 10

12) **LOWER()**:- Returns a specified character expression in lower-case letters.

**Syntax :-** *LOWER(<expC>)*

e.g. ?LOWER("DATA POINT")

Return:- data point

13) **LTRIM()**:- Returns the specified character expression with leading blanks removed.

**Syntax :-** *LTRIM(<expC>)*

e.g. ?'FoxPro programming is easy',LTRIM(" friends")

Return:- FoxPro programming is easy friends")

**14) REPLICATE():-** Repeat the character expression at given number.

**Syntax :-** *REPLICATE(<expC>, <expN>)*

e.g. ?REPLICATE('\*',4)

Return:- \*\*\*\*

**15) RIGHT():-** Returns the specified number of rightmost characters from a character string.

**Syntax :-** *RIGHT(<expC>, <expN>)*

e.g. ?RIGHT("data point",3)

Return:- int

**16) RTRIM():-** Returns the specified character expression with all trailing blanks removed.

**Syntax:-** *RTRIM(<expC>)*

e.g. :- ?"HELLO FRIENDS" ,RTRIM("WEL-COME                    "), " TO LEARN" FOXPRO""

Return:- HELLO FRIEND WEL-COME TO LEARN

**17) STUFF():-** Returns a character string by replacing a specified number of characters in a character expression with another character expression.

**Syntax :-** *STUFF(<expC1>, <expN1>, <expN2>, <expC2>)*

e.g. ?STUFF("I have a bigger elephant",10,6,"big")

Return:- I have a big elephant

Here, <expC1>:- Is the string in which you want to replace the character.

<expN1>:- is the starting position of text.

<expN2>:- is the number of character upto which you want to replace the character.

<expC2>:- is the new text which you want to put in string.

**18) SUBSTR():-** Returns a specified number of characters from the given character expression or field.

**Syntax:-** *SUBSTR(<expC>, <expN1>[, <expN2>])*

e.g. ?SUBSTR("Foxpro is DBMS package",11,4)

Return:- DBMS

Here, <expC> :- The character expression from which characters are extracted is .

<expN1> :- starting position of character , where character extraction begins.

<expN2>:- specifies the number of characters to be extract from the character expression.

**19) TRIM():-** Returns the specified character expression with all trailing blanks removed. TRIM( ) is identical to RTRIM( ).

**Syntax:-** *TRIM(<expC>)*

e.g. :- ?"HELLO FRIENDS" ,TRIM("WEL-COME                    "), " TO LEARN"

Return:- HELLO FRIENDS WEL-COME TO LEARN

**20) UPPER():-** Returns the specified character expression in upper-case.

**Syntax:-** UPPER(<expC>)

e.g. :- ?UPPER("data point")

Return:- DATA POINT

## FILE MANIPULATION COMMANDS

1) **COPY TO** :- Copies the contents of the current table/.DBF to a new file.

**Syntax:-**

*COPY TO <file>*

*[FIELDS <field list>]*

*[<scope>]*

*[FOR <expL1>]*

*[WHILE <expL2>]*

*[TYPE][ WK1 | WKS | WR1 | WRK | XLS |*

e.g.

☞ Copy only those records which have city as SHEGAON and salary >3000

*USE EMP*

*COPY TO XYZ FOR CITY="SHEGAON" AND SALARY>=3000*

☞ Copy only fields name,city,dob,course

*USE STUD*

*COPY TO XYZ FIELDS NAME,CITY,DOB,COURSE*

2) **COPY STRUCTURE** :- Copies the structure of the current table/.DBF to a new table/.DBF.

**Syntax:-**

*COPY STRUCTURE TO <file>*

*[FIELDS <field list>]*

e.g.

☞ Copy entire structure to xyz

*USE STUD*

*COPY STRUCTURE TO XYZ*

☞ Copy only fields name,dob,course

*USE STUD*

*COPY STRUCTURE TO XYZ FIELDS NAME,DOB,COURSE*

3) **COPY FILE**:- Create a duplicate file of any file.

**Syntax:-**

*COPY FILE <file1> TO <file2>*

e.g.

☞ Copy the Stud.Prg file into Student.Prg

*COPY FILE STUD.PRG TO STUDENT.PRG*

☞ Copy Xyz.Dbf into Xyz1.Dbf

*COPY FILE XYZ.DBF TO XYZ!.DBF*

*Tip:- when DBF file will copy the file must be closed.*

4) **RENAME**:- Change the name of original file into new name.

**Syntax:-**

*RENAME <file1> TO <file2>*

e.g. Change the name *Xyz.Dbf* to *Abc.Dbf*

*RENAME XYZ.DBF TO ABC.DBF*

5) **DELETE FILE:-** Delete any file from memory.

**Syntax:-** *DELETE FILE<filename>*

e.g.- *DELETE FILE XYZ.DBF*

*tip:- When you delete the database/.dbf file the file must be closed. & you must specify the extension of file.*

6. **SCAN...ENDSCAN:-** Moves the record pointer through the current table/.DBF and executes a block of commands for each record that meets the specified conditions.

**Syntax:-**

*SCAN*

*[<scope>]*

*[FOR <expL1>]*

*[WHILE <expL2>]*

*[<statements>]*

*[LOOP]*

*[EXIT]*

*ENDSCAN*

7) **DO CASE.... ENDCASE:-** Executes the first statement block after DO CASE and before END CASE whose associated conditional statement evaluates to true (.T.).

**Syntax:-**

*DO CASE*

*CASE <expL1>*

*<statements>*

*[CASE <expL2>*

*<statements>*

*...*

*CASE <expLN>*

*<statements>]*

*[OTHERWISE*

*<statements>]*

*ENDCASE*

8) **DO WHILE...ENDDO:-** Executes a block of statements within a conditional loop.

**Syntax:-**

*DO WHILE <expL>*

*<statements>*

*[LOOP]*

*[EXIT]*



*ENDDO*

9) **EXIT:-** Exits a DO WHILE, FOR or SCAN loop.

**Syntax:-**

*EXIT*

10) **FOR...ENDFOR:-** Executes the commands after FOR and before ENDFOR within a loop a specified number of times.

**Syntax:-**

*FOR <memvar> = <expN1> TO <expN2> [STEP <expN3>]  
    <statements>*

*[EXIT]*

*[LOOP]*

*ENDFOR | NEXT*

e.g.

FOR mcount = 1 TO 10

? mcount

ENDFOR

11) **IF...ENDIF:-** It is Conditional statement which executes a set of commands based on the logical expression.

**Syntax:-**

*IF <expL>*

*<statements>*

*[ELSE*

*<statements>]*

*ENDIF*

e.g.

CLOSE DATABASES

USE customer

LOCATE FOR city="AKOLA"

IF FOUND( )

? 'Company: ' + company

ELSE

? 'Condition ' + temp + ' was not found '

ENDIF

USE

12) **FOUND():-** Returns true if CONTINUE, FIND, LOCATE or SEEK is successful.

**Syntax:-**

*FOUND([<expN> | <expC>])*

e.g. USE STUD

STORE 0 TO mcount

LOCATE FOR COURSE = 'ADCSSAA'

DO WHILE FOUND( )

mcount = mcount + 1

CONTINUE

ENDDO

WAIT WINDOW 'Total student form ADCSSAA course: ' ;

+ LTRIM(STR(mcount)) NOWAIT

13) **INPUT:-** Inputs data from the keyboard into a system memory variable or an array element.

**Syntax:-**

*INPUT [<expC>] TO <memvar>*

e.g. Input "Enter Number A" To A

?A

14) **ACCEPT:-** Accepts character string data from the screen.

**Syntax:-**

*ACCEPT [<expC>] TO <memvar>*

e.g. Accept "Enter the name:-" to name

15) **@... GET:-** Creates an editing region.

**Syntax:-**

*@ <row, column> GET <memvar> | <field>*

*[FUNCTION <expC1>]*

*[PICTURE <expC2>]*

*[FONT <expC3>[, <expN1>]]*

*[STYLE <expC4>]*

e.g. @12,2 get name pictur "@!" font "arial",16 style "BI"

read

16) **READ:-** Activates objects created with @ ... GET and @ ... EDIT commands.

17) **@... SAY:-** Displays output at a specified row and column position.

**Syntax:-**

*@ <row, column> SAY <expr>*

*[FUNCTION <expC1>]*

*[PICTURE <expC2>]*

*[SIZE <expN1>, <expN2>]*

*[FONT <expC3>[, <expN3>]]*

*[STYLE <expC4>]*

e.g. @12,2 say "WEL-COME TO DATA POINT" style "BI"

18) **@...GET -(Check Box):-** You can create the control buttons. To create check box you required the numeric variable

**Syntax:-** *@<row>,<col> get mvariable FUNCTION "\*C <titleof button>"*

e.g. STORE 0 TO CH

@3,4 GET CH FUNCTION "\*C ADD"

Return:- ADD { }

*Tip:- you can use PICTURE instead of FUNCTION like @3,3 get ch PICTURE "@\*C ADD" the value of check box is 1 when it is checked otherwise 0. to define hot key "</>" sign is used before the character.*

19) **@...GET-(Radio button):-** Radio button required numeric or character variable.

**Syntax:-** @<row>,<col> get mvariable FUNCTION "\*R<title>;<title>;.."

e.g. STORE 0 TO CH

@3,4 GET CH FUNCTION "\*R \<ADD;D\<ELETE"

Return:- ( ) ADD

( ) DELETE

20) **@...GET-(Push Butt.):**- Same as radio button .

**Syntax:-** @<row>,<col> get mvariable FUNCTION "\*R<title>;<title>;.."

e.g. STORE 0 TO CH

@3,4 GET CH FUNCTION "\* OK;CANCEL"

Return:- <OK >

<CANCEL>

*tip:- If the button wants to display horizontally the use H with \* like*

@3,4 GET CH FUNCTION "\* H OK;CANCEL" it display like <OK> <CANCEL>  
by default it is vertically placed.

21) **@...GET-(Popup):**- To create the popup control you need numeric or character variable. You use ^ symbol as the function code or @^ as picture code.

**Syntax:-** @<row>,<col> get mvariable FUNCTION "^ <title>;<title>;.."

e.g. STORE 0 TO CH

@3,4 GET CH FUNCTION "^Graduate;MBA;BA"

*tip:- to activate the check box,radio button,popup,push button READ or READ CYCLE command is used.*

### **What is memo field? How it works?**

Memo field used to store the long textual information. Suppose we considered that you need to create a *dbf* having information as book\_name, authour\_name, & book\_summary. Here we store book\_name,authour\_name in character field, but we can't use the character field for summary, because character field can't be wider than 254 character. So you use the *MEMO*

field for storing the summary.

When a memo field used with in database file ,the actual information in memo field stored in auxiliary database file. Foxpro automatically create the auxiliary file (.fpt) when you create the memo field in dbf.

For example if you create a database file like *data.dbf* having *memo field* foxpro create two files *data.dbf* & *data.fpt* which is auxiliary file.

## ADVANTAGE of MEMO FIELD:-

- ✓ A memo field occupies only 10 bytes of space in database record. You can store any amount of text for each record.
- ✓ As actual data for the memo field stored in auxiliary file, the size of main *dbf* Reduced.

### Entering Data In Memo Field

To entered data in memo field, ensure that the cursor is positioned in the memo field & press ^Home or ^Pgdn. If you have a mouse you can double-click on “memo”.

This opens the window for entering the text for memo field for the record. After you entered the information press ^W to close the window. Or select *File –close from foxpro menu.*

Foxpro returns you to *edit/append/browse* window, now you see that “memo” will change to “Memo” . *Memo* indicate that it contain the data whereas *memo* indicate that it can't contain the data.

In this way you can entered the data in memo field.

### Listing Memo Field

When you give *LIST command* to display the content of field on screen, it can't display the content from *Memo field*. To display it you must type the name of that field

e.g. *list b\_name,a\_name,summary*. By default width of memo field is 50 character. This default width can be changed through the *SET MEMOWIDTH* command.

For e.g. *SET MEMOWIDTH TO 25*  
*LIST OFF B\_NAME,A\_NAME,SUMMARY*

### What Is Report & Labels In Foxpro.

Using *LIST* command you can't give the presentable output form also it don't have the facility to include page title,page number,date,headings etc. you can't also apply the format to the text. But foxpro gives the facility as *REPORT* to create the presentable output form.

### Features Of Foxpro Reports:

- ✓ A page title can be given to the report.

- ✓ Any field expression can be placed anywhere in detail line.
- ✓ Descriptive text can also include in the page.
- ✓ Report title can be print on every page.
- ✓ The report can have the page footer that print- text,field,variable,function to display the information.
- ✓ The data in the report can be grouped.
- ✓ Print style(bold,italic,underline,color) can be given to text.
- ✓ Calculation such as sum,average,maximum,minimum can be performed on numeric field.

### ***How To Create The Report***

To create the report CREATE REPORT command is used.

Tip:- before creating report open the data base file for which you want to create the report.

Syntax:- CREATE REPORT <NAME>

*When you give the command CREATE REPORT command it open the window with the given name & extension .frx. for e.g. DATA.FRX*

*The screen divided into three parts :-*

Pghead:- *used to display the report titled,column heading or description.*

Detail:- *used to display the field information that will actually form the report.*

Pgfoot:- *used to display the footer information that will appear on each page.*

### **Creating Quick Report**

*When you give the CREATE REPORT command you will see the REPORT menu is added to the foxpro menu. For quick report :-*

- 1) *Click on Report menu it display the menu options .*
- 2) *From this select Quick Report it display the report layout(Column,Form layout) select any one & click on OK.*

*Now you will see the format on your window.*

### **Previewing The Report**

*Foxpro enable you to preview the report before printing.*

*Click on REPORT menu then select PAGE PREVIEW.*

### **Saving The Report**

*To save the report press ^W or select Save form File menu.*

### **Generating The Report**

*Now you can generate the report using REPORT FORM command you can specify the report format file .*

e.g. USE STUD

REPORT FORM STUDENT

*You can also see the report in page-by-page view. To use this mode add the clause PREVIEW to REPORT FORM command.*

For e.g. REPORT FORM STUDENT PREVIEW

### **Printing The Report**

*You can also print the report on printer by adding the clause TO PRINTER to REPORT FORM command. It display report on screen as well as printer.*

For e.g. REPORT FORM STUDENT TO PRINTER

*Now your printer print the report. If printer on then it print the report otherwise give the error message. If you don't want to see the report on screen give NOCONSOLE as*

e.g. REPORT FORM STUDENT TO PRINTER NOCONSOLE

### **Printing Selected Report**

*You can also print the selected report using FOR clause. As*

REPORT FORM STUDENT FOR CITY ="KHAMGAON"

### **Saving The Report Into Text File**

*The report generated with the REPORT FORM command can be save the record in other text file.*

For e.g. USE STUD

REPORT FORM STUDENT TO FILE STUD.TXT

### **Creating The Custom Report**

*You can also create the custom report. Steps to create the report:-*

1. *Open the DBF file & give the command CREATE REPORT STUDENT*
2. *It display the window in PgHead give the title of report as "Mark sheet of student".*
3. *Then move the cursor to next line & give the column name as NAME, COURSE CF,CP,FOXPRO,SAD,ORACAL TOTAL,PER,RESULT,GRADE.*

4. Then move the cursor to the **DETAIL** line & press **^F** to open the field expression box again press **ENTER** button. Now it display the **Report Expr. box** select the field name from field list at left side. & select **OK**. Again select **OK**.

Tip:- you can press **^W** instead of selecting **OK** button.

5. In this way you can select the other fields which you want.

6. Then move the cursor to **PgFoot** & type **DATE:-** to display current date press **^F** to display the field expre. box & again press **ENTER** button. From this select the **Date()** from date function. & press **^W** two times.

7. Now to see the report press **^I** or select **REPORT-PAGE PREVIEW**.

### Shortcut Key In Report

Key	Meaning
1. ^I	Page Preview
2. ^F	Open field expression box.
3. ^O	Remove the line
4. ^N	Add New Line
5. ^B	Create Box
6. ^G	Bring to front
7. ^J	Send to back]

### Grouping Of Data In Report

You can group the data in report. For exam. suppose you want to group the employees from same city to get the grouped the report ,you need to arrange the record of database in same order.

STEPS:-

1. first open the database & index it as  
**USE EMPL**

**INDEX ON CITY TO CITY**

2. Then give the command **CREATE REPORT EMPC** it opens the window.

3. Give the title **CITYWISE EMPLOYEE INFORMATION** & give the heading of columns as you want .(like **NAME,DOB,DOJ,BASIC,NETPAY**).

4. Then move the cursor to **DETAIL BAND** & press **^F** to add field & again press **ENTER** it display the fields select one & press **^W** two times.

5. In this way you can create the fields .Now select **REPORT** menu & select **DATA GROUPING**. It show the box select **ADD** push button it display another dialog box press here **ENTER** & select the field **CITY** which you index. &press **^W** three times. Now you will see on Line is displayed above & below the detail band like **1-City**. Type there **CITY:-** & press **^F** & press **ENTER**. Select **CITY** & press **^W** two times.

6. Now move cursor to **pgfoot** & give footer information & see the report preview by pressing **^I**. You will see the grouped data by city wise.

## CREATING LABEL:-

*Foxpro provide another facility for designing & printing the mailing label. The extension of .LBX*

*To create label format file CREATE LABEL command is used. When you create label you must open the dbf file for which you create the label.*

*Steps:-*

1. *Open the database file as USE EMPLOYEE*
2. *Then give the command CREATE LABEL EMPL*
3. *It display the window here the default size is 3 ½" X 15/16" X 2 you can also change it manually.*
4. *then move the cursor to the WIDTH & give the width .*
5. *move cursor to Number Across- it means how many labels placed in one row .*
6. *move cursor to Space Between – it indicate that how many space left between the labels.*
7. *move cursor the Lines Between- it indicate that the vertical distance between two label.*
8. *you can set the height of label.*
9. *to entered the field expression press ^F it display the box select the field & press ^W.*
10. *In this way you can add number of fields in label , if you add the self text/prompting then type the text in " " symbol.like "Name:-".*
11. *Then to see the preview press ^I. To save the label press ^W.*

## Generating The Label

*Now you can generate the label using LABEL FORM command you can specify the report format file.*

*e.g. USE STUD*

*LABEL FORM STUDENT*

*You can also see the report in page-by-page view. To use this mode add the clause PREVIEW to LABEL FORM command.*

*For e.g. LABEL FORM STUDENT PREVIEW*

*You can save the label to another text file as*

*USE STUD*

*LABEL FORM STUDENT TO FILE MYFILE.TXT*

*You can also print the label to printer as*



USE STUD

LABEL FORM STUDENT TO PRINTER

*You can print the selected records as*

USE STUD

LABEL FORM STUDENT FOR RESULT ="PASS"

**Explain the difference between EXIT & LOOP command in do while...enddo**

*The normal procedure to exit the DO WHILE ...ENDDO statement is when the specified condition becomes false.*

### **EXIT**

There is another way to terminate the loop using EXIT command within DO WHILE ...ENDDO statement. When foxpro encountered the EXIT command ,no more commands from the loop is executed. It exit the loop & executed the statement next to the ENDDO statement. The EXIT command is normally used in IF ...ENDIF statement.

*For e.g.*

*Store 0 to x,sum,y*

*Do while x<=5*

*Input "Enter No." to y*

*If y<0*

*Exit*

*Else*

*Sum=sum+y*

*Endif*

*X=x+1*

*?sum of no. is',sum*

### **LOOP**

LOOP is another statement used in DO WHILE ..ENDDO statement. LOOP is specified within IF ...ENDIF. When foxpro encountered the LOOP command,the control pass to the DO WHILE statement,the remaining commands( upto ENDDO ) can't executed. It again check the condition & execute the command ,until the condition becomes false

*For e.g.*

*store 0 to x,y,sum*

*Do while x<=5*

*Input "Enter The No." to y*

*If y<0*

*LOOP*

*else*

*sum=sum+y*

*endif*

*x=x+1*

*?sum of no. is',sum*

**Difference between IIF() & IF statement**

IF is decision making statement. These are of two forms IF..ENDIF & IF...ELSE..ENDIF. The first statement check the condition & if it is true execute the command , otherwise jump out off loop. Where as second if condition is TRUE the true statement is executed otherwise FALSE statement is executed.

*The format is:-*

*If <condition>*

*...*

*<command>*

*...*

*endif*

*for e.g. consider the following program*

INPUT "ENTER YOUR SALARY" TO SALARY

IF SALARY>3000

? "YOUR SALARY IS MORE THAN 3000"

ENDIF

When you run this program foxpro prompts you if you entered the salary greater than 3000 otherwise it display no message.

IIF() – The IIF() function, that is also same as IF..ELSE..ENDIF. This function return one expression if the condition true, return the TRUE statement otherwise FALSE statement. Both these expression specified within IIF().

*Syntax:- IIF(CONDITION,TRUE STATEMENT,FALSE STATEMENT)*

*For e.g.*

*?IIF(salary>3000,"salary more than 3000","salary less than 3000")*

*if the condition is true then , "salary more than 3000" is executed otherwise*

*"salaryless than 3000" executed.*

### **Difference between FOR ...ENDFOR & SCAN ...ENDSCAN**

FOR...ENDFOR executes a set of statements within a loop a specified number of times. A memory variable or array is used as counter to specifiy how many times the statement executed inside the loop. Consider the following prog.

STORE 0 TO NUM

FOR NUM- 1 TO 10

?SQRT(NUM)

ENDFOR

By default FOR...ENDFOR increments the variable by 1. However, if you want to increase/decrease the variable +/- sign is used after STEP option like STEP -3 or STEP 3.

SCAN ...ENDSCAN:- Moves the record pointer through the current DBF file & execute the block of commands for each record that meets the specified condition.

*Syntax:-*

SCAN [SCOPE]

[FOR <expL>]

[LOOP]

[EXIT]

ENDSCAN

*For e.g.*

USE address

ACCEPT "ENTER CITY:-" TO MCITY

SCAN FOR CITY=MCITY

?NAME

?CITY

?DIST

?

ENDSCAN

### **Difference between INDEX & SORT**

#### **INDEX**

1. It arranges the record in index file.
2. It requires less memory as it arranges /store only the record number.
3. The extension of index file is .IDX (Simple index) & .CDX (Compound index) file.
4. The new data automatically updates

#### **SORT**

1. It sorts the records in new DBF file.
2. It requires same memory as the Original database.
3. The extension is .DBF
4. New data will not be updated.

when added.

**Syntax:-**

INDEX ON [field,field,...] to [idx file]

TAG [cdx file][ascending/descending]

For e.g

Use stud

Index on rollno tag roll

**Syntax:-**

SORT TO [filename] ON [field,...]

[FOR <expL>]

For e.g.

use stud

sort to stu on rollno

### ***Explain how to generate screen.***

You can also create the screen using *CREATE SCREEN* command. Screen builder is most useful facility in foxpro. In this you just place the text, fields & variables on the screen at desired position. You can easily move the object on the screen. You can also format the screen. Once you are satisfied with the appearance you may generate the screen.

**STEPS:-**

- ✓ Open the data base for which you want to create screen.
- ✓ Give the command *CREATE SCREE STUD*.
- ✓ It opens the window, for quick screen you select *SCREEN—QUICK SCREEN* or you can press ^F to add field. It displays the field expr. box, to input the text select <Get..> or to display list select <Say..> then select the field & press ^W two times.
- ✓ In this way you can select the fields you want.
- ✓ To generate the code select *PROGRAM—GENERATE* option from it displays the screen generate box select the appropriate option & select *GENERATE*. After this press ^W & to execute the screen give the command

*DO STUD.SPR*

It displays the screen.

### ***What is RQBE? Explain it with example?***

RQBE means Relational Query By Example. Query means the process of extracting specified information from file/table. You can relate two different tables/files to create new file.

To create query file *CREATE QUERY <NAME>* is used. The extension of query file is *.QRY*. You can order the data by using *ORDER BY* option. By using *GROUP BY* option you can create group of data. You can also create the specified condition using *HAVING* option. To run/execute the query *DO <name.ext>* command is used. For e.g. *Do Stud.Qry*.

**STEPS:-**

1. Open the data base to which you want to create the query. *USE STUD*
2. Give the command *CREATE QUERY STUD* it opens the window.

3. Then select *ORDER BY* to order/index the record. It display the window select the fields which you want & select ascending/descending option & select ok.

4. Then select *HAVING BY* it display the window, under field name *select the field* (for e.g *CITY*) & move the cursor to next tab select here the option from drop down list(for e.g. *LIKE*),then move the cursor to next tab type here the condition(for e.g. *"SHEGAON"*).( If you want to extract the record for *city "shegaon" or salary greater than 3000*. then you use OR option & then move tab to field name select salary field then move cursor to next tab select here more than option ,then move the cursor to next & type 3000. ) select OK.

5. To see the query select *DO QUERY* option from window. It display the list in browse form.

*Tip:- instead of OR you can use AND operator. You don't specify the operator you just select the next field.*

To save the query press ^W. To execute the query give command *DO STUD.QRY* in command window.

### **What is TEXT-ENDTEXT?**

When you need to display the text use *???* instead of this you can use *TEXT-ENDTEXT* to display long message. You don't require to use quotations ,you can type several lines. You can also print the text on *PRINTER*.

For e.g.

@2,1 say "Display The menu"

TEXT

1 Add Record

2 Modify Record

3 Delete Record

4 Exit

ENDTEXT

The above command display the matter on screen from text..endtext.

### **Explain the CALCULATE command?**

Calculate is multipurpose command that you use to perform various mathematical function. The function include sum,average,min,max etc.

*Syntax:-*

*CALCULATE<exprlist>*

*[<scope>] [FOR <expL>] [TO <memvar list>]*

e.g. USE SALES

CALCULATE SUM(PROFIT) TO TPROFIT

?TPROFIT

Calculate the total of profit field & stor it into TPROFIT field.

### **What is structural compound index file?**

A compound index file can hold more than one index file in a single file. Each index file within a compound index file is called index tag. The compound index file given a extension .CDX It is also called *Structural Compound* index file. It has same name as the opened .DBF file with extension .CDX . You can also create separate compound

index file which is called *Independent Compound* index file (with extension .CDX) but this file will not open automatically when DBF file open. The records can not reindex whenever you will not open the independent compound index file.

For e.g.

USE STUD

INDEX ON NAME TAG NAME

It create structural compound index file with name STUD.CDX.